Universidad Nacional de La Matanza

Florencio Varela 1903 - San Justo - Buenos Aires - Argentina

Departamento de Ingeniería e Investigaciones Tecnológicas

Sistemas Operativos

Jefe de Cátedra: Fabio Rivalta

Autor: Guido Gariup

Año: 2013

Apunte sobre RAID

(Redudant Array of Inexpensive Disks)

Este resumen está basado, en gran parte, en el siguiente documento de "The PC Guide":
http://www.pcguide.com/ref/hdd/perf/raid/index.htm
Por otro lado, también se ha extraído información del blog "VMdamentals" (www.vmdamentals.com), específicamente de los artículos "Throughput Part 1" hasta "Throughput Part 4" y "RAID5 DeepDive and Full-Stripe Nerdvana".

RAID: Es el acrónimo de Redundant Array of Inexpensive Disk, aunque actualmente se ha reemplazado la palabra "Inexpensive" por "Independent".
La idea fundamental detrás de esto es contar con un conjunto de discos, llamado Array, que se comportan como si fuera uno solo.
Existe lo que se denominan diferentes niveles de RAID los cuales se utilizan de acuerdo a las necesidades de la aplicación o del usuario, los cuales explicaremos más adelante en este mismo documento.
El concepto de RAID se puede implementar tanto en software como en hardware, siendo este último el más utilizado últimamente, además de brindar mayor performance.
Este tipo de tecnología hace algunos años sólo estaba disponible para ambientes corporativos debido a su alto costo, y si uno quería implementar una solución de este tipo tenía que terminar optando por la solución de RAID por software. Hoy en día los costos han disminuido de tal manera que muchos de los motherboards a los que hoy tenemos acceso como usuarios ya disponen de controladoras RAID integradas en el hardware.
Pero antes de adentrarnos en el concepto de RAID, recorramos un poco la historia y algunos otros conceptos por demás interesantes.
Debido a que el concepto de RAID trata sobre la agrupación de discos que albergan la información, comencemos preguntándonos dónde están los discos. ¿A qué nos referimos con esta pregunta? Nos referimos a en qué lugar uno pone los discos...sí, sí, los discos físicos, esos discos rígidos que uno sabe que están casi siempre en los sistemas de computación. Uno intuitivamente piensa que en el único lugar donde podrían colocarse los discos en las computadoras hogareñas tiene que ser dentro del gabinete de la computadora (mal llamado CPU). Sin embargo, ¿se han preguntado alguna vez dónde se colocan esos discos en los servidores o sistemas de cómputo que generalmente se encuentran en los datacenters?
En un principio, en los servidores también se pueden colocar discos rígidos. Pero pregúntense qué pasaría si ese servidor donde están los discos que contiene la información por alguna razón fallase (se quemara el motherboard, el microprocesador, la fuente de alimentación, etc.). En un principio uno podría pensar que simplemente podría sacar los discos del servidor y colocarlos en otro. Sin embargo, cuando nos vamos de las PCs hogareñas hacia los servidores de marca (IBM, HP, Cisco, Dell, etc.) generalmente estos fabricantes diseñan sus propios discos para sus propios servidores. En realidad, los discos físicamente la mayoría de las veces son los mismos, pero lo que se denomina el "enclosure" (o la carcasa o soporte) no lo es. A continuación se muestran algunas imágenes de discos para servidores de los diferentes fabricantes del mercado para que se den cuenta de qué estamos hablando.


Disco HP


Disco IBM

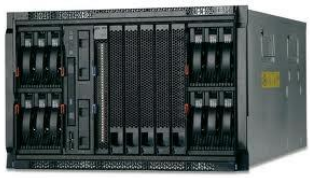

Disco Oracle (ex Sun)

Tecnologías para la interfaz de conexión de los discos existen muchísimas variedades. Las más conocidas son IDE, SCSI, SAS, SATA, NL-SAS y FC. Las que hoy más se utilizan en el mercado de datacenter son SAS, NL-SAS y SATA.
Y también mostramos algunos servidores de los diferentes fabricantes para que vean también de qué estamos hablando.

Servidor HP Rackeable

Servidor Cisco Rackeable

Servidor IBM Tower



Servidor Cisco Blade

Servidor IBM Blade con Storage incorporado (Blade Center S)

Como verán de las figuran anteriores, no es posible colocar discos de una marca en servidores de otra, o por lo menos no directamente. Por lo tanto, volviendo al punto anterior, si un servidor que contiene discos queda fuera de servicio, también queda afectada la disponibilidad de la información que en los discos reside.
Por tal motivo, hace muchos años se crearon los sistemas de almacenamiento externos para albergar la información de los servidores. A estos sistemas en el ambiente se los denomina “storage”, y es así que los denominaremos en este apunte.
De esta manera, los storage nos permiten alojar ahí los discos, en uno o varios chasis y los diferentes servidores acceden a la información como si estuvieran accediendo directamente a discos conectados internamente a ellos. A continuación mostramos algunas imágenes de diferentes storage del mercado.

Línea EMC VNX (5100 a 5700)

IBM DS3524 (24 discos de 2,5”)

HP P4000 (12 discos de 3,5”)

Por lo tanto, ahora la información reside en los discos que están en los storage, y si ahora un servidor quedara fuera de servicio, la información no se vería afectada ya que el storage es totalmente independiente de los servidores. De hecho, cualquier otro servidor que estuviese conectado al storage podrá acceder a la información. Claro está, ahora hemos trasladado el problema del único punto de falla al storage, pero para ello estos sistemas permiten la utilización de tecnologías de replicación, de tal manera que automáticamente, sincrónica o asincrónicamente estos sistemas replican parte o toda la información de un sistema a otro.
Pero nos encontramos con otro problema que no lo teníamos antes cuando los discos estaban conectados directa e internamente dentro del servidor, que es que ahora debemos resolver la cuestión de la comunicación entre los servidores y los storage. Y cuando de comunicación hablamos inmediatamente se nos vienen a nuestra mente los protocolos. De tal manera, que a lo largo del tiempo se han ido desarrollando diferentes protocolos para dicha comunicación.
Pero antes de definir los protocolos, vamos a definir un par de conceptos básicos:

- **SAN (Storage Area Network):** La SAN es una red de almacenamiento que conecta a los servidores con los storage. Según el tipo de tecnología utilizada (tipo de protocolo) la SAN puede ser una red totalmente separada físicamente de la red LAN que todos conocemos o bien, compartir la red LAN y quedar separada “logicamente”.
- **HBA (Host Bus Adapter)**: Genéricamente hablando, la HBA es la interfaz con la que se conectan los servidores a la red SAN para llegar a los storage. Otra vez, según el tipo de protocolo y tecnología esta HBA podrá ser una placa física que se conecta en el servidor (por ejemplo las

HBA para Fibre Channel que veremos a continuación), una placa de red común y corriente (en el caso de los protocolos de almacenamiento que comparten la red LAN) o por último, una CNA.

- **CNA (Converged Network Adapter):** Las redes convergentes de datos y almacenamiento son la tecnología que hoy día se está impulsando. En estas redes se envían al mismo tiempo tráfico de red común y corriente y tráfico de almacenamiento. Estas placas generalmente se presentan al sistema operativo como placas de red y al mismo tiempo como placas HBA para almacenamiento. Es decir, una misma placa física, dentro del sistema operativo se ve como si fueran dos. Estas placas manejan el protocolo de almacenamiento FCoE que lo veremos a continuación.
- **Controladoras (o Storage Processors o simplemente SPs):** Son las encargadas de llevar a cabo toda la operatoria dentro de los storage. Son las que tienen la "inteligencia" de estos sistemas. La mayoría de éstos vienen en una configuración dual-controller, lo que significa que tienen dos controladoras para brindar redundancia. Estas controladoras, al igual que lo hacen las HBA, también se conectan a la SAN para quedar de esta manera conectados con los servidores. Según el storage, las controladoras pueden trabajar en modo activo-activo o activo-pasivo. En este último caso, la controladora pasiva o standby sólo entra en servicio cuando la primera falla. En cambio en los storage activo-activo ambas se encuentran dando servicio al mismo tiempo, repartiendo la carga entre las mismas, además de ofrecer, claro está, capacidad de redundancia o failover.
- **Latencia:** En redes la latencia es sinónimo de demora. En el ambiente de almacenamiento hay muchísimas latencias a medir (latencia interna del kernel del sistema operativo, que mide cuánto tiempo el comando SCSI generado por el SO tarda en entregarse a la HBA, latencia interna de la HBA, que mide cuánto tiempo tarda el comando SCSI en salir al medio físico, etc.). La latencia que más nos interesa generalmente es la latencia física total, es decir, el tiempo que tarda desde que se genera un comando SCSI dentro del sistema operativo hasta que llega la respuesta por parte del storage (es decir, sumando la latencia interna del SO, la de la HBA, el tiempo que tarda en transmitirse el comando por la red física y todo su camino de vuelta).

A continuación mostramos algunas imágenes de las tarjetas CNA.

CNA Física

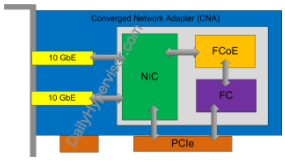


Diagrama lógico de una CNA

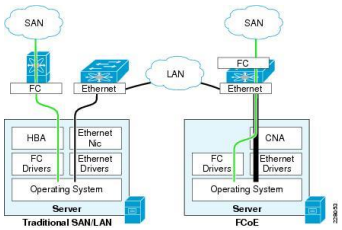


Diagrama de red tradicional y de red convergente con CNA

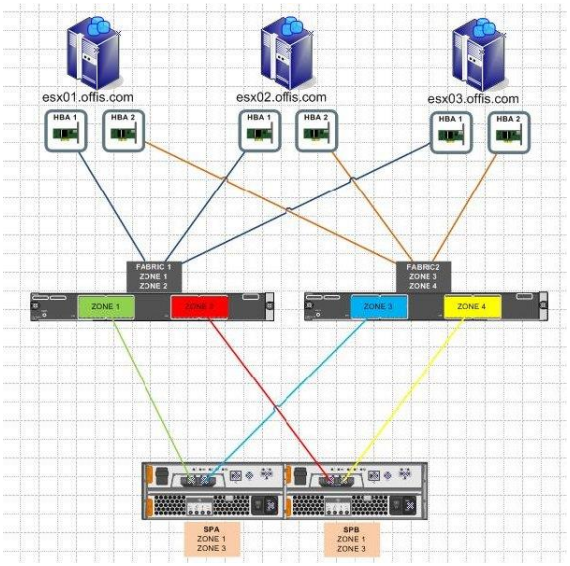
Fíjense del diagrama lógico (segunda figura), cómo internamente por el bus PCIe del servidor, el mismo envía el tráfico de red común y corriente (NIC) y el tráfico Fibre Channel de almacenamiento, el cual es encapsulado en FCoE (Fibre Channel Over Ethernet) y enviado al medio físico. En la última imagen se muestra un diagrama de red convencional y otro convergente. Observen cómo en las redes tradicionales el flujo de información de red común y corriente sale del servidor por una placa de red tradicional (NIC) y el flujo de información de almacenamiento sale por una HBA específicamente destinada a tal fin. En cambio, en las redes convergentes una única placa (la CNA) maneja los dos flujos de información al mismo tiempo. Por supuesto, estas redes necesitan de equipos especiales (switches especiales) para también poder manejar la información de manera convergente.

Ahora sí, volvemos a los protocolos de almacenamiento. Estos protocolos de comunicación entre servidores y storage se dividen en dos grandes grupos:

- **Protocolos orientados a bloques**. ¿Qué significa que un protocolo sea orientado a bloques? Significa que los servidores cuando quieren acceder a algún dato simplemente envían al storage un comando SCSI indicando el número de bloque a leer/escribir (hay diferentes versiones de SCSI; SCSI-2, SCSI-3, etc). Este comando viaja hacia el storage, alguna de las controladoras del

storage lo agarra, va al o a los discos donde se encuentra dicho bloque y realiza la operación. A vuelta de correo, el storage le contesta al servidor con otro comando SCSI indicándole la respuesta al comando original. El tema es cómo viaja ese comando. Para eso se desarrollaron los diferentes protocolos que se detallan a continuación (los más utilizados):

- **FC o Fibre Channel**. Básicamente el comando SCSI se encapsula en un frame Fibre Channel y dicho frame Fibre Channel se entrega a la HBA del servidor y la HBA físicamente (generalmente con haces de luz porque se utiliza fibra) hace llegar el comando al storage. La vuelta es similar. Lo importante acá es que cuando tenemos dos servidores no hay problema porque conectamos las HBAs directamente a las controladoras del storage (generalmente cada controladora tiene dos puertos por lo que permite que dos servidores se conecten en forma redunante). El problema aparece cuando tenemos un tercero, cuarto, quinto, etc. ¿Qué hacemos? Tenemos que poner algo en el medio, y ese algo es un switch FC, que al igual que los switches ethernet, se utilizan para concentrar la red en algún punto y hacer que varios servidores se puedan conectar con el mismo storage. A esos switches en la jerga de FC se los conoce como “Fabric”.



En el diagrama vemos una interconexión típica de una red SAN Fibre Channel. Fíjense cómo cada servidor cuenta con dos placas HBA para brindar redundancia. Cada HBA de cada servidor se conecta a un switch FC (fabric). Y a su vez, cada controladora del storage (SP A y SP B) también se conecta a cada switch. De esta manera, toda la solución es totalmente redundante frente a la falla de cualquier HBA de los servidores, de cualquiera de los dos switches o incluso frenta a la falla de una de las controladoras del storage.
Al igual que en las redes LAN había una manera de segmentar dicha red en forma lógica, utilizando las VLANs (redes virtuales) en las redes Fibre Channel existen el concepto de “zonning” o zonificación. El zonning permite que se definan zonas de trabajo y hace que solamente los equipos que pertenecen a la misma zona se puedan ver entre ellos. De esta manera, aunque los switches sean los mismos, mediante el uso de las zonas puedo configurar que ciertos servidores vean determinados storage y otros servidores vean a otros storage, dando seguridad al sistema y evitando que un equipo vea a otro que no tiene por qué ver.

- **iSCSI o IP SCSI**. Los mismos comandos SCSI del punto anterior en lugar de encapsularse en un frame Fibre Channel se pueden encapsular en un segmento TCP, después en un paquete IP y después en un frame Ethernet. ¿Con qué sentido? Que si tenemos un frame ethernet podemos utilizar los mismos switches ethernet que todos ya tenemos. Por lo tanto eso es iSCSI. Ahora la SAN (Storage Are Network) la armo utilizando los mismos equipos que ya vengo utilizando para interconectar a las computadoras y servidores de mi red. ¿Tengo que tener algo

en cuenta? Debería. Por ejemplo debería considerar seriamente dedicar una VLAN (Virtual LAN) sólo para el tráfico iSCSI, de tal manera que los broadcast que se generan en mi red no me perjudiquen a los frames iSCSI (no se olviden que esto está simulando una SAN y la latencia la quiero mantener lo más baja posible). Por lo tanto, si utilizo la misma red física para transportar el tráfico de red común y corriente y el tráfico de almacenamiento, al menos tengo que separar lógicamente una red de otra (por eso la recomendación del uso de VLANs). Por otro lado debería considerar habilitar los Jumbo Frames, frames que exceden el tamaño estándar de una red LAN (1500 Bytes) de manera tal de bajar lo más posible el overhead creado por el múltiple encapsulamiento en las diferentes capas.

- **FCoE o Fibre Channel Over Ethernet**. A medida que avanzaba el tiempo, otro equipo de trabajo pensó que en lugar de pasar por la capa de transporte y la capa de red, podíamos agarrar el mismo comando SCSI y encapsularlo en un frame Fibre Channel, luego en un Frame Ethernet y seguir utilizando switches ethernet de red. Sin embargo ahora no me es posible seguir utilizando los mismos switches de red que venía utilizando anteriormente. Esto es debido a que Fibre Channel no es un protocolo que esté preparado para perder tramas (frames). En cambio Ethernet es lo que se denomina un protocolo de mejor esfuerzo, es decir, hago lo mejor posible para que las tramas lleguen a su destino, pero si alguna se pierde, alguien tendrá que retransmitir la información. ¿Por qué entonces puedo utilizar iSCSI que a su vez utiliza los switches ethernet comúnes y corriente y no lo puedo hacer con FCoE? Sucede que iSCSI en la primera capa en que se encapsula es la de transporte, utilizando el protocolo TCP. En el mundo del TCP/IP, este trabajo de control sobre los segmentos lo realiza justamente el TCP. Sin embargo, FCoE no utiliza TCP (como sí lo hacía iSCSI), por lo tanto, ¿quién es el que me va a brindar seguridad? Si no lo hacen los protocolos, entonces lo debe hacer el hardware. Ergo, para poder utilizar FCoE se debe contar con switches Ethernet especiales que utilizan la tecnología de Data Center Bridging, la cual permite asegurar que el tráfico Fibre Channel que viene encapsulado en Ethernet llegue sin problemas a destino, sin pérdida de frames. A estos switches de los conoce con el nombre de Lostless (sin pérdida).
- **SAS**. Éste es un híbrido entre la tecnología SAN y Direct-Attach (es decir, discos conectados directamente en el servidor). No es ni una SAN pura ni tampoco son discos directamente conectados (se comporta como una SAN, pero la tecnología que se utiliza es la de los discos conectados directamente). Sólo van a ver sistemas de storage con conectividad SAS en IBM, HP y LSI. ¿Qué es lo que hicieron estas marcas? Agarraron el bus interno SAS que se utiliza dentro del servidor y lo extendieron fuera del mismo. Pusieron mecanismos de arbitraje dentro de las controladoras para que varios servidores pudieran acceder a los datos al mismo tiempo (sin "chocar" entre ellos) y lograron utilizar la velocidad SAS de 3Gbps a un costo muy accesible. Lo bueno es que se obtiene muy buenas performance a un costo accesible. Lo malo es que las controladoras tienen cuatro puertos cada uno. Pasa lo mismo que con Fibre Channel pero con 4 servidores. Cuando queremos poner el quinto servidor hay que poner un switch. El problema es que no son muy comunes estos swithes. Donde se los ve más a menudo es en los servidores de tipo blades. Como decíamos anteriormente, a estas redes con estos storage no se las considera una SAN en el sentido más puro de la definición (ya que no utilizan ninguno de los protocolos mencionados anteriormente) pero brindan el mismo servicio como si lo fueran (es decir, permiten que varios servidores accedan a un mismo storage al mismo tiempo). Por eso decimos que son un híbrido. Se las considera direct attach.

- **Procotolos orientados a archivos:** ¿Qué es que un protocolo sea orientado a archivo? Significa que en lugar de que el servidor ejecute un comando SCSI sobre el storage lo que hace es hacer una consulta de un determinado archivo en una determinada ruta. Por ejemplo, "leer el archivo /etc/passwd". A los equipos que trabajan en esta modalidad se los conoce como NAS (Network Attach Storage). Los dos protocolos más conocidos son:

- **NFS:** Network File System. Desarrollado por Sun Microsystems para su sistema SUN Solaris. Se convirtió en un estándar para la compartición de archivos en sistemas UNIX. En la jerga de NFS el directorio a compartir se dice que se "exporta".
- **CIFS:** Common Internet File System. Aunque muchos en este momento se estén preguntando qué es esto, casi todos lo usaron alguna vez. Cada vez que un usuario en Windows ingresa en el cuadro de texto para ejecutar \\dirección_IP_Servidor en Windows para acceder a una carpeta compartida que se encuentra en otra máquina, están haciendo uso de este protocolo. Es también conocido como Samba o SMB. Es el protocolo desarrollado por Microsoft para la compartición de archivos e impresoras en sistemas Windows y también se lo conoce como Samba porque es el *daemon* que se levanta en sistemas Linux para acceder a carpetas compartidas en Windows (suena bastante paradójico, ¿no? A un protocolo de Microsoft de lo conoce más por el nombre de un servicio que se utiliza en Linux que por el nombre que Microsoft le dio). Originalmente este protocolo está basado en SMB, el cual fue inventado y desarrollado por IBM.

Es importante aclarar bien el funcionamiento de estos dos tipos de protocolos ya que mucha gente dice "si al fin y al cabo alguien va a terminar ejecutando un comando SCSI para leer la información de los discos, ¿cuál es la diferencia entre los dos tipos de protocolos, orientados a bloques y a archivos?". La diferencia radica en quién es el que ejecuta el comando SCSI y qué es lo que viaja por la red. En los protocolos orientados a bloques, el sistema operativo que se ejecuta en el servidor "ve" a los discos que están en el storage como si tuvieran acceso directo a ellos y ejecutan directamente los comandos SCSI. Es decir, en los protocolos orientados a bloques los storage le presentan a los servidores unidades lógicas (llamadas LUN – Logical Unit Number) que los servidores "ven" como discos propios. Y por lo tanto, los servidores ejecutan los comandos SCSI sobre estas LUNs directamente, y estos comandos son los que viajan por la red.
En los protocolos orientados a archivos la información que viaja por la red como peticiones son comandos del propio protocolo (comandos definidos en NFS o en CIFS) con sus diferentes parámetros, entre ellos la ruta y el archivo a acceder. Es decir, el sistema operativo que se ejecuta sobre el servidor sabe que la información a la que está accediendo es un directorio exportado o una carpeta compartida, según el protocolo, y envía las órdenes para acceder a ellos a la controladora del storage, y es esta última la que ejecuta el comando SCSI para acceder a la información.
Ahora que hemos hecho la introducción a los sistemas de almacenamiento y la forma de acceder a ellos nos adentraremos en el mundo RAID.

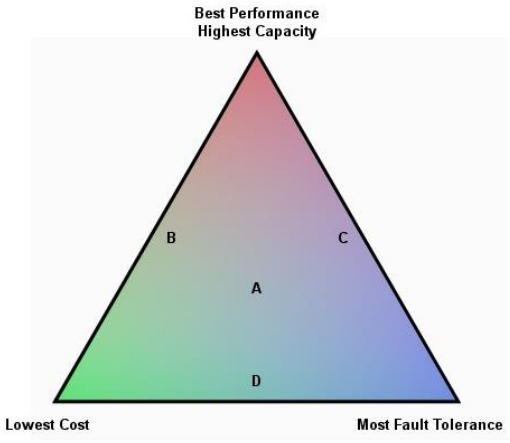
¿Por qué usar RAID?

Básicamente se utiliza un RAID por varios motivos. Los principales son: mayor confiabilidad y disponibilidad de los datos frente a eventuales fallas de hardware (principalmente rotura de discos), mayor performance y mayor capacidad que utilizar un disco solo.
Veamos algunos de los beneficios de utilizar esta tecnología:

- **Mayor seguridad en los datos:** Básicamente este beneficio viene dado por la utilización de cierta redundancia al escribir los datos en los diferentes discos, de manera tal que frente a la falla total de un disco (a veces más según el nivel de RAID utilizado) los datos igualmente se puedan recuperar. Todos los niveles de RAID proveen cierto tipo de protección de datos excepto RAID0.
- **Tolerancia a fallos:** Está muy ligado con el punto anterior. Gracias a la escritura de datos redundantes (excepto en RAID0) se logra una mayor confiabilidad de todo el sistema de almacenamiento que al utilizar simplemente un disco individual.
- **Disponibilidad mejorada:** Los sistemas con RAID mejoran la disponibilidad de los datos gracias a la tolerancia a fallos y características especiales que permiten la recuperación frente a fallos sin caída del sistema, transparente para el usuario o sistema.
- **Capacidad incrementada:** Gracias a tratar a todos los discos como "uno", la capacidad total se incrementa con respecto al uso de un solo disco individual.

- **Performance mejorada:** En ciertos niveles de RAID, el uso de múltiples discos mejora la performance de escritura y lectura con respecto a la performance obtenida al utilizar un solo disco.

La tecnología de RAID, como muchas tecnologías, se basan en lo que se conoce como "Essential Tradeoff Triangle". Este triángulo tiene en cada uno de sus vértices al "Menor Precio", "Mayor Performance" y "Mayor Tolerancia a Fallos". Básicamente lo que indica esta analogía es que sólo se pueden tener al mismo tiempo dos opciones de las descriptas. Se puede tener un sistema RAID que posea una muy alta performance y que sea muy tolerante a fallos, pero no será barato. Por otro lado se puede tener un sistema que tenga muy alta performance y que sea barato, pero sacrificaremos fiabilidad en el mismo. Y por último, podemos tener un sistema muy confiable, muy barato pero a costa de una pérdida importante en su performance.



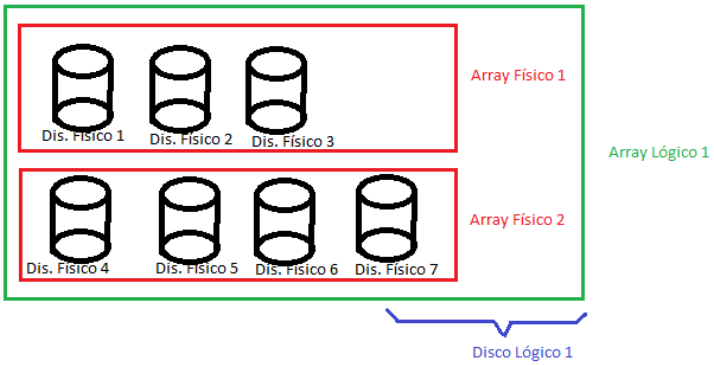
Es muy importante hacer notar que si bien los sistemas RAID proveen, en su mayoría, alta disponibilidad gracias a la redundancia, estos sistemas NO solucionan todos los problemas aparejados con la pérdida de datos. Por ejemplo, un sistema RAID no provee ninguna solución frente a un virus, o frente al borrado casual o intencional de datos por parte de usuarios, etc., por lo que se debe tener en cuenta que estos sistemas NO son un reemplazo de los sistemas de backup de información.

Niveles de RAID

Antes de hablar de los diferentes niveles de RAID, definamos algunos conceptos:

- **Discos físicos:** Son los discos físicos que todos conocemos.
- **Arreglo o Array físico:** Es el conjunto de discos físicos. Uno o más discos físicos unidos conforman un arreglo o array físico de discos.
- **Arreglo o Array lógico:** Los arreglos o array lógicos se conforman particionando un array físico o bien combinando más de un array físico en un array lógico. Por ejemplo, se podría tener un arreglo físico de 10 discos físicos, y a partir de él crear dos arreglos lógicos conformados cada uno por ejemplo, de 5 discos físicos cada uno. Así también se podría tener dos array físicos de 3 discos físicos cada uno, y formar a partir de ellos un solo array lógico conformado por la totalidad de los 6 discos. En la gran mayoría de los casos prácticos, esta relación es de 1 a 1, es decir, se crea un arreglo lógico por cada arreglo físico.
- **Discos lógicos:** Los discos lógicos se crean dentro de los array lógicos y se pueden crear más de un disco lógico por cada array lógico. En definitiva, estos discos son los que "ve" el sistema operativo como uno solo. Estos discos lógicos en la mayoría de los sistemas de almacenamiento se los conoce como LUN (Logical Unit Number)

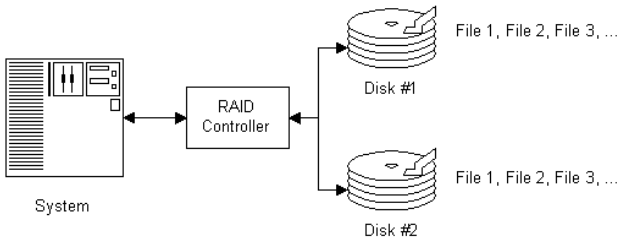
Fíjense que esta versatilidad de conceptos nos permitiría tener por ejemplo, algo así como mi disco *"/dev/sda"* dentro del Linux, el cual en realidad es un disco lógico, que es sólo una parte de un array lógico que está conformado a su vez por dos array físicos de 3 y 4 discos físicos respectivamente.



Para el ejemplo que recién dimos, el disco lógico N° 1 sería el que sistema operativo Linux “ve” como “/dev/sda”.

Mirroring

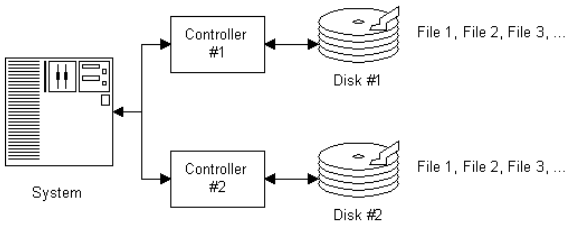
El concepto de *Mirroring*, tal como la palabra lo indica, se basa en que el sistema escribe en dos discos la misma información. La ventaja principal del mirroring es que no sólo provee una completa redundancia de los datos sino que además provee una muy rápida recuperación frente a fallos ya que al fallar uno de los discos, la información ya se encuentra en el segundo de ellos inmediatamente.
Gráficamente el esquema es como se muestra a continuación:



Esta técnica de mirroring es utilizada en RAID1 como así también cualquier nivel de RAID que involucre el nivel 1, por ejemplo RAID10.

Duplexing

El concepto de *Duplexing* es el mismo que para el *mirroring* sólo que además también se duplica la controladora de discos.
El esquema para este caso sería el siguiente:

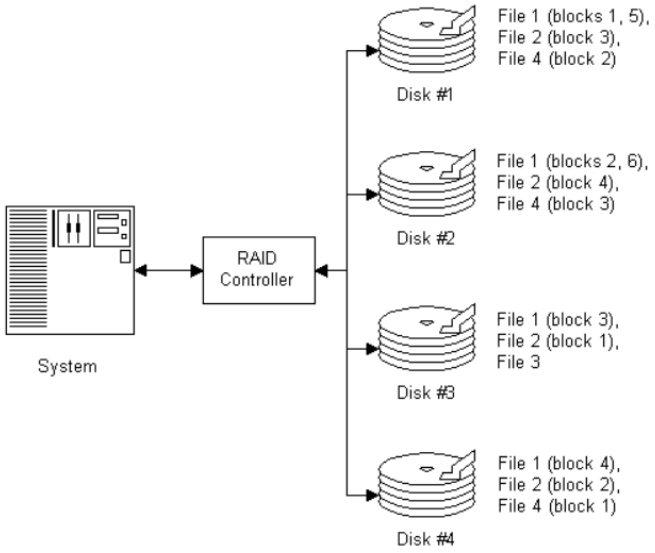


La mayoría de las soluciones de storage (almacenamiento) del mercado de hoy en día permiten adquirirlos en una configuración con controladora redundante o lo que se llama en la jerga, *dual-controller.* Esta configuración no es más que la aplicación de este concepto de *duplexing*.
Dichas controladoras, en los equipos que se conocen como *entry-level* o más económicos trabajan en una configuración Activa/Pasiva. Esto quiere decir que para el acceso a un determinado disco lógico sólo hay una de las dos controladoras activas, y frente a una eventual falla de ésta, la pasiva toma el control.
En los equipos de más alta gama, ambas controladoras trabajan como Activa/Activa, queriendo decir que para un mismo disco lógico ambas controladoras están activas y esto permite a los sistemas enviarle

las peticiones de entrada/salida (IO) a ambas controladoras a la vez, permitiendo así un reparto de carga entre ambas controladoras.

Striping

El *striping* es una técnica que básicamente viene a luchar contra la gran limitación de performance que sufren los discos físicos al tener que posicionar sus partes mecánicas para la escritura/lectura de datos. Debido a que los sistemas RAID por lo general tienen muchos discos que conforman un array físico, lo que el *striping* busca es sacar ventaja de la escritura o lectura simultánea en más de un disco a la vez.
El *striping* se puede realizar a nivel de bytes o de bloques. El *striping* a nivel de bytes significa que a un archivo se lo divide en partes de tamaño de un byte. La primera parte es enviada al primer disco, la segunda al segundo y así sucesivamente. Muchas veces también se utiliza como tamaño de esas piezas el tamaño de los sectores del disco (comúnmente 512 bytes).
El *striping* a nivel de bloques lo que hace es dividir al archivo en partes de igual tamaño y cada una de ellas es enviada a los diferentes discos. Por supuesto, cuando se envió una determinada parte al último de los discos, la siguiente parte del archivo vuelve a enviársele al primer disco.
Gráficamente esto se vería de la siguiente manera:



Esta técnica se utiliza en la mayoría de los niveles RAID. En RAID0 se utiliza *striping* a nivel de bloques sin paridad, en RAID 3 y 7 se utiliza *striping* a nivel de bytes con paridad y en RAID 4, 5 y 6 *striping* a nivel de bloque con paridad.
Es muy importante notar que el *striping* no provee por sí solo redundancia de datos. Para ello se utiliza la paridad o el mirroring como ya vimos.
Un concepto que se define aquí es el *stripe*, que es el conjunto de bloques que se escriben en todos los discos que conforman el array físico. Para la figura anterior por ejemplo, el primer stripe sería el bloque 1, 2, 3 y 4 del archivo 1, el segundo stripe sería el bloque 5 y 6 del archivo 1 más el bloque 1 y 2 del archivo 2, y así sucesivamente.

Parity

Escribir datos redundantes mediante la técnica de *mirroring* es relativamente fácil, pero tiene dos desventajas. La primera y más importante es que se pierde el 50% de la capacidad total, ya que por cada disco que utilizo para guardar datos, necesito otro igual para guardar los mismos datos duplicados.
Por otro lado, no provee mayores ventajas en lo que a performance se refiere, ya que no se utiliza en esta técnica el *striping.*
Otra manera de guardar información redundante es el uso de la paridad. La paridad hace uso de una propiedad matemática de la operación binaria lógica XOR, u “O-Exclusivo”. Esta operación binaria tiene la propiedad que arroja un valor verdadero cuando sí y sólo sí uno de los dos operandos es verdadero.
Comparemos la tabla de verdad de la operación “OR” con la de la operación “XOR”.

Input		Output	
#1	#2	"OR"	"XOR"
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Lo mejor de esta operación es que si uno calcula por ejemplo A XOR B, nos arroja un resultado que si a su vez lo volvemos a operar con la misma operación con cualquiera de los dos operandos, nos da como resultado el otro operando. Por ejemplo:

A XOR B = C

Si ahora hacemos:

C XOR B = A

O bien:

C XOR A = B

Con esto podemos concluir que si tenemos una determinada cantidad (N) de partes (A, B, C, etc.) y aplicamos un XOR a cada una de ellas de la siguiente manera:

(((A XOR B) XOR C) XOR etc.) = P

El resultado P es justamente la paridad. Luego, si guardo cada una de las partes más la paridad en N + 1 discos, lograré que en dicho conjunto de discos (array físico) yo tenga la información con la redundancia suficiente para que ante cualquier falla de uno de ellos, pueda recuperar la información.
Como ejemplo hagámoslo con sólo dos partes y tres discos. Tenemos por un lado la parte A de la información y la parte B de la misma información. Al querer escribirla, calculo la paridad P de la siguiente manera:

A XOR B = P

Luego, en el disco 1 escribo la parte A, en el 2 la parte B y en el último la paridad P. Ahora, frente a la falla de cualquier disco, tengo la posibilidad de recuperar la información. Por ejemplo, si me falla el disco de paridad no hay problema alguna, ya que la información está completa en los primeros dos discos. Si llega a fallar el primer disco, el que contiene la parte A, simplemente calculo qué valor tenía A de la siguiente manera:

B XOR P = A

De igual forma, calcularía el valor de B como A XOR P.
Esta técnica de utilizar la paridad en conjunto con el striping me permite ahora disponer de redundancia en los datos frente a una eventual falla en alguno de los discos y además mejorar mi performance porque ahora escribo en más de un disco al mismo tiempo. Por otro lado, ahora ya no desperdicio más el 50% de mi capacidad total, ya que el espacio utilizado para albergar la paridad es (100/N)%. Para una configuración con 3 discos, el espacio desperdiciado es del 33%, para una con 4 discos es del 25%, etc.
Pero por otro lado ahora tengo los siguientes problemas. Ahora cuando escribo un dato en el array, tengo que hacer un cálculo de paridad que antes con el mirroring no era necesario. Por eso, es muy importante disponer de controladoras de alta capacidad de cómputo ya que el retardo introducido por dicho cálculo impactará directamente en la performance.
Además, como contra tiene que la recuperación ante fallos ya no es tan transparente como en la técnica de mirroring, ya que si el disco que falla NO es el de paridad, ahora cada vez que tenga que recuperar un valor del disco que falló tengo que también hacer el cálculo de paridad.
A partir de RAID3 hasta RAID7 todos los RAID utilizan la paridad como medio de introducir redundancia en la información. RAID2 utiliza un método similar pero no es exactamente el mismo.

Vale aclarar a esta altura que los principales vendors del mercado de los sistemas de storage (IBM, HP, EMC, NetApp, etc.) generalmente desarrollan sus propias técnicas de introducción de redundancia en los datos, que si bien son generalmente más eficaces que los estándar, son técnicas propietarias de cada vendor.

Temas relacionados a la performance en ambientes RAID

Un sistema RAID se sustenta en el "paralelismo" para poder brindar mejores performances que un solo disco individual, donde el acceso se realiza a un solo disco y no a "muchos".
Se podría hablar muchísimo de performance en sistemas RAID que no es el caso, pero sí vale la pena repasar algunos conceptos importantes para tener en cuenta por qué algunos RAID performan mejor que otros.
Ya en los sistemas basados en discos individuales, las funciones de lecturas Vs. escrituras performan de diferente manera. Cuando nos vamos a un sistema basado en RAID, esta diferencia se magnifica.
La diferencia fundamental de un sistema RAID a la hora de escribir información es que cuando se escribe información con redundancia, se debe acceder a todos los lugares físicos en donde la información se va a alojar. En cambio, cuando se lee, sólo hay que acceder a aquellos lugares que contienen en sí la información, pudiendo obviar la lectura de la información redundante.
Veamos cómo aplica esto a las diferentes técnicas explicadas anteriormente:

- **Mirroring:** La lectura con este tipo de técnicas es superior a la escritura. Esto se debe a que cuando se escribe información, dicha información debe ser escrita al mismo tiempo en dos discos en lugar de uno. Como mucho, el tiempo de escritura podría ser igual al que se obtiene al utilizar un solo disco individual. Sin embargo, cuando hablamos de lectura, ahora la información sólo se debe leer de uno de los dos discos, pudiendo el otro estar atendiendo a otra petición de lectura al mismo tiempo.
- **Striping sin redundancia:** Se obtiene aproximadamente la misma tasa de escritura que de lectura.
- **Striping con paridad:** Acá nuevamente la performance de escritura es peor a la de lectura, pero a diferencia del mirroring, acá la penalidad es más sustancial. A continuación algunas cosas a tener en cuenta para esta técnica:
    - **Para las lecturas:** En realidad la lectura con *striping* con paridad puede resultar un poco más lenta que el caso de striping sin paridad ya que al leer, el bloque de paridad no hace falta leerlo, por lo que se comportaría como un RAID0 (striping sin paridad), con la salvedad que en este último la información se dispersa en un disco más aumentando el paralelismo para leer la información. ***NOTA:*** esto que escribo acá difiere en un sentido totalmente opuesto a lo que dice el documento, el cual asegura que la lectura con striping con paridad puede resultar más veloz que sin paridad debido a que se utiliza un disco más para albergar los datos (cosa que no es cierto porque justamente se utiliza un disco menos, debido a que el restante se utiliza para la paridad). Pienso que debe ser un error de tipeo y donde dice "faster" en el documento debería decir "slower".
    - **Para escrituras secuenciales:** Hay un doble overhead ya que además de sumarse el tiempo para el cálculo de la paridad, hay que escribir en un disco dicha paridad.
    - **Para lecturas/escrituras aleatorias:** Acá ocurre la mayor discrepancia entre la lectura y escritura. Las lecturas que necesiten de sólo una parte del stripe de uno o dos discos pueden procesarse en paralelo con otra lectura que necesite otra parte del stripe en otro disco, haciendo que la lectura sea realmente muy veloz. El problema aparece con las escrituras, y esto se debe a que frente a cualquier escritura en una parte del stripe requiere de dos escrituras más la lectura de las demás partes del stripe para calcular la paridad. Esto se ve muy fácil con el siguiente ejemplo. Consideremos un array físico en RAID5 conformado por los discos 1, 2, 3, 4 y 5, donde para un stripe en especial los datos están en los discos 3, 4, 5 y 1, y la paridad en el disco 2. Si se tuviera que escribir sólo el bloque o la parte del stripe perteneciente al disco 3, si no se tuviera paridad se escribiría sólo ese bloque y listo. Pero al tener la paridad, el cambio en el bloque del disco 3 implica tener que leer los bloques del disco 4, 5 y 1 para poder con ellos calcular la paridad (sumando tiempo), y luego escribir el bloque en sí de datos del

disco 3 más la nueva paridad calculada en el disco 2. Es por esto que cuando se tienen aplicaciones que generan muchas escrituras aleatorias, por ejemplo muchas de las bases de datos, el RAID5 no es para nada una buena opción.
Y por último, otra cuestión que genera mayores tiempos en las escrituras son los discos dedicados a albergar la paridad, en particular las implementaciones con RAID3 ó 4. Debido a que por cada escritura se tiene que escribir una nueva paridad, ese disco dedicado a albergarla termina convirtiéndose en un cuello de botella. Este problema se vio aplacado con el uso del nivel 5 de RAID, donde la paridad no se aloja sólo en un disco sino que por cada stripe la paridad se aloja en un disco diferente.
Y como si todo esto fuese poco para lidiar con la escritura en sistemas con striping con paridad, hay que sumarle la dificultad de que la controladora del disco debe hacer la escritura de los datos y de la paridad simultáneamente sin interrupción, ya que se podría si no dar el caso que sólo se escriba el dato y si el proceso en ese momento se interrumpe, los datos en ese stripe quedarían corruptos (ya que la paridad no reflejaría el verdadero valor que debe tener).

- **Casos especiales de escrituras aleatorias:** Como vimos en el caso anterior, "lecturas/escrituras aleatorias", nos damos cuenta que la escritura particularmente tiene una gran penalidad en los niveles de RAID que utilizan striping con paridad. No obstante analicemos algunos casos para ver que esto puede ser muy perjudicial (siempre refiriéndonos a la performance, claro está) o tal vez no tanto.
  En el caso anterior analizábamos la gran pérdida de performance que se tenía cuando teníamos que escribir sólo el segmento 3 de un stripe formado por 5 segmentos (o bloques). En ese caso, el sistema operativo intentaba ejecutar un IO (comando de entrada/salida) de escritura, pero en total se generaban 5 IOs (3 para leer los segmentos 1, 4 y 5; y 2 para escribir el nuevo segmento 3 más la nueva paridad calculada en el segmento 2). Cuando hablamos de penalidad en la escritura, decimos que en este caso tenemos una penalidad de 5 (5 IOs necesarios para ejecutar sólo 1).
  No obstante démonos cuenta de que existe otra forma de "zafar" para no tener una penalidad tan alta. Recordemos que el problema se suscita porque al escribir un sólo segmento esto hace que la paridad de todo el stripe se modifique y es por eso que de alguna manera tenemos que calcular la nueva paridad que contemple el nuevo valor del segmento a escribir (en este ejemplo que venimos siguiendo el segmento 3). Ahora bien, si en lugar de leer todos los demás segmentos del stripe (en el ejemplo el 1, 4 y 5) leemos el valor actual que tiene el segmento a escribir (el 3) y leemos la paridad actual (el 2), junto con el nuevo valor del segmento a escribir (el 3 nuevo) podemos calcular la nueva paridad sin necesidad de leer "todos" los demás segmentos.
  Veámoslo con un ejemplo para que nos quede más claro. Utilizaremos para el mismo simplemente un bit para indicar el valor de cada segmento del stripe, pero es exactamente lo mismo si fuera un bit, 512 ó 64 KB). Supongamos que tenemos el siguiente stripe en un momento dado:

  1**0**1**1**00

  En este caso la paridad (en el segmento 2) es la que está marcada en color verde, y los datos son los valores de los segmentos 1, 3, 4 y 5. En este caso los datos son 1100 y la paridad se calcularía como 1 XOR 1 XOR 0 XOR 0 que da, como está en el ejemplo, cero. El dato a cambiar es el del segmento 3 que está en color rojo. Actualmente el valor de ese segmento es 1. Supongamos ahora que el sistema operativo quisiera escribir un 0 en ese segmento. Si en lugar de seguir la metodología de leer todos los demás valores aplicamos la nueva metodología de leer el valor actual del segmento a escribir más la paridad y luego recalcular el valor de la nueva valiéndonos del nuevo valor a escribir tendríamos:
  0 XOR 1 XOR 0 = P' = 1 (P' es el nuevo valor de la paridad para el stripe, el 0 en verde es el valor de la paridad actual, el 1 en rojo es el valor del segmento 3 actual y el 0 en azul es el nuevo valor a escribir en el segmento 3). Por lo tanto con el nuevo procedimiento y la nueva paridad calculada, el stripe completo quedaría conformado como:

  1**1**0**0**0

  Donde ahora el valor 1 de la paridad (en color verde) es el calculado en el paso

anterior y el 0 en el segmento 3 (de color azul) es el nuevo valor escrito en el mismo. Si nos fijamos con cuidado, los comandos de IO que se ejecutaron ahora para llegar al mismo resultado son 4 (2 para leer el valor actual de los segmentos 2 (la paridad) y 3 (valor viejo del segmento); más 2 para escribir los nuevos valores en los segmentos 2 (paridad nueva, P') y 3(nuevo valor que se quería escribir). Por lo tanto, ahora obtuvimos una penalidad de 4. Hemos mejorado, pero sin embargo seguimos teniendo una penalidad importante. Lo interesante es que esta penalidad de 4 ahora es constante, no importa cuántos segmentos haya en el stripe (piensen cuánto hubiera sido la penalidad utilizando el método anterior que daba 5 si los segmentos del stripe en lugar de haber sido 5 hubieran sido 8 ó más). Con esta nueva metodología, no importa si la cantidad total de segmentos del stripe es 5, 8 ó más ya que siempre los pasos a seguir son los que enumeramos a continuación:

1. Leer el segmento a ser modificado (el 3 en el ejemplo)
2. Leer el segmento que contiene la paridad (el 2 en el ejemplo)
3. Recalcular la nueva paridad (porque tenemos el valor anterior del segmento a modificarse, el valor anterior de la paridad y el nuevo valor del segmento a modificarse)
4. Escribir el nuevo valor en el segmento a modificarse (el 3 en el ejemplo)
5. Escribir el nuevo valor de la paridad recalculada en el paso 3.

¿Y qué pasaría ahora si en lugar de modificar un sólo segmento quisiera modificar dos? Siguiendo con este nuevo método lo que debería hacer es:

1. Leer los dos segmentos a modificar
2. Leer el segmento de paridad
3. Recalcular la nueva paridad
4. Escribir los dos nuevos segmentos
5. Escribir la nueva paridad recalculada en el paso 3

Como verán, ahora para escribir dos segmentos (dos IOs) se necesitan de 6 IOs (2 para leer los valores anteriores de los segmentos a modificar, 1 para leer el valor anterior de la paridad, 2 para escribir los nuevos valores en los segmentos y 1 para escribir la nueva paridad). Por lo tanto, para el caso de la escritura de dos segmentos del stripe, ahora la penalidad de escritura es de 3 (6 IOs necesarios para 2 IOs a escribirse).
Si seguimos el mismo método para escribir ahora 3 nuevos segmentos, nos daremos cuenta que los IOs necesarios a ejecutar son 8, por lo que ahora la penalidad de escritura es 2,66. Y así sucesivamente.
Ahora fíjense si el sistema fuera lo suficientemente inteligente (y los storage modernos lo son) para poder combinar este último método con el primero (el que leía los segmentos que no se modificaban). En este último caso, en lugar de leer los tres segmentos a modificarse, simplemente leería el único segmento que *NO* se modificará. Este segmento junto con los tres a modificarse se pueden utilizar para el cálculo de la nueva paridad. Por lo tanto, se tendrían los siguientes pasos:

1. Leer el único segmento que no se modifica
2. Calcular la paridad (junto con los tres segmentos a escribir)
3. Escribir los nuevos tres segmentos
4. Escribir la paridad calculada en el paso 2

Por lo tanto, ahora para escribir 3 segmentos (3 IOs) se necesitó de la ejecución de 5 IOs (1 para leer el segmento no modificado, 3 para escribir los nuevos segmentos y 1 para escribir la paridad), lo que da una penalidad de 1,66 (5 IOs necesarios para 3 IOs a escribirse) . Este número incluso es mejor que la penalidad que se obtiene en un RAID10 (muchas veces pensado como el mejor RAID para todos los casos) que tiene una penalidad de escritura constante de 2 (ya veremos más adelante que este nivel de RAID siempre tiene esta penalidad ya que debe escribir todo por duplicado).
Sin embargo, lo mejor ocurre cuando tenemos la necesidad de escribir en todos los segmentos del stripe. En este caso, no hay necesidad de leer ningún segmento anterior, y por lo tanto los pasos en este caso son:

1. Calcular la paridad (ya que todos los datos a escribir son nuevos y ya los tenemos)
2. Escribir los cuatro nuevo segmentos
3. Escribir la paridad calculada en el primer paso

Como podrán observar, ahora para poder ejecutar 4 IOs se necesitan de 5 IOs, lo que da una penalidad de escritura de 1,25.
De todo este desarrollo, podemos sacar como conclusión que la técnica de paridad junto con striping es la más performante de toda *siempre y cuando* se realice una escritura completa del stripe. Sin embargo, es un hecho que en la mayoría de los ambientes modernos se utiliza la virtualización, y en éstos la mayoría de las escrituras son pequeñas y aleatorias, con lo que es muy difícil que se escriban stripes enteros.
Entonces, ¿para qué sirve esta técnica de escribir todo el stripe al mismo tiempo si en la mayoría de los casos no la voy a poder utilizar? Acá es donde viene la ayuda de las memorias cache. Si estas memorias se utilizan para la escritura, el storage puede demorar la escritura de los diferentes segmentos hasta que los mismos conformen un stripe entero y recién ahí volcar físicamente los datos a los discos.
Alguien podría preguntarse qué sucede si hay un corte de energía eléctrico y hay datos en estas memorias cache sin volcarse a disco aún. La respuesta es que la mayoría de los storage modernos que utilizan estas memorias cache cuentan con baterías internas en las controladoras que permiten mantener por bastantes horas los datos en estas memorias hasta que el suministro eléctrico se restablezca y las mismas puedan volcar los datos a los discos. Y cuando estas baterías han cumplido con su ciclo de vida y hay que reemplazarlas, los sistemas desactivan el uso de las memoria cache para escritura.

Performance asociadas al posicionamiento y a la transferencia

En este apartado básicamente se definen dos conceptos cuando se tiene que acceder a un disco, tanto para lectura como para escritura.

- **Tiempo de posicionamiento:** Aquel tiempo que se emplea para que la cabeza del disco se sitúe en la posición adecuada para leer o escribir un dato.
- **Tiempo de transferencia:** Aquel tiempo que se emplea para transferir un dato desde o hacia el disco una vez que la cabeza está posicionada.

Definidos estos dos conceptos podemos realizar un pequeño estudio de cómo se ven alterados ambos tiempos según la técnica utilizada:

- **Mirroring:** Durante una lectura sólo se utiliza uno de los dos discos, pero la controladora puede utilizar ambos discos para realizar dos operaciones de lecturas independientes. Por lo tanto, esto disminuye el tiempo de posicionamiento. Sin embargo, cuando el dato a leer se encuentra, la lectura se realiza de un solo disco. Dicho esto se concluye que para las lecturas secuenciales no habrá una mejora en la perfomance. Para las escrituras se utilizan ambos discos y generalmente la performance es peor que al utilizar un solo disco individual.
- **Striping:** Los archivos grandes que se dividen en varios bloques, y por lo tanto en varios discos, requieren que todos los discos se posicionen antes de leer los datos, por lo tanto el tiempo de posicionamiento no se ve mejorado. Sin embargo, cuando las cabezas de los discos están posicionadas, los datos se leen todos al mismo tiempo mejorando en gran medida el tiempo de transferencia. Las lecturas de archivos pequeños que no requieren leer la información de todos los discos, permiten que la controladora atienda varias peticiones de lectura al mismo tiempo, con lo que se ve mejorado el tiempo de posicionamiento, no tan así el de transferencia. Los tiempos de escrituras aleatorias, como ya vimos, generalmente se ven degradados debido al cálculo de la paridad.

¿Qué es más importante, el tiempo de posicionamiento o de transferencia? No hay una respuesta tajante a esta pregunta, pero como una regla rápida se puede decir que para archivos grandes es mejor

tener una buena performance en la transferencia y para archivos pequeños tener una buena performance en el tiempo de posicionamiento.
Mucha gente sobrevalora el gran incremento en la performance que se obtiene en las tasas de transferencias secuenciales al utilizar striping, pero para la mayoría de los casos o mejor dicho, para el promedio de los usos, esto no representa realmente una gran ventaja.

Ancho y tamaño del stripe

Hay dos parámetros con que los sistemas de RAID permiten trabajar. Uno es el ancho del stripe (stripe width) y el otro es el tamaño (stripe size).
El ancho del stripe se define como la cantidad de segmentos que se pueden escribir o leer simultáneamente de los discos, y como es lógico, concuerda con la cantidad de discos. Es decir, si tenemos 4 discos, el ancho de nuestro stripe será de 4.
A mayor ancho de stripe mayor performance, ya que se aumenta el paralelismo. Esto quiere decir que se obtendrá una mayor performance con 8 discos de 18 Gb. que con 4 de 36 Gb.
El segundo parámetro, stripe size, también es conocido como block size, chunk size, stripe length o granularity. Se refiere al tamaño de cada uno de los segmentos que se escriben o leen en un disco. Este tamaño sí es configurable por el usuario y típicamente se configura entre los 2 Kb y los 512 Kb en potencias de 2. Por ejemplo, si se tiene un array de 4 discos con bloques en cada uno de ellos de 512 Kb se tendrá un stripe con width 4 y size 512 Kb. Muchas veces se comete el error de decir que el stripe size es el tamaño de cada uno de los bloques multiplicado por el stripe width. Para el ejemplo anterior, erróneamente se diría que el stripe size es de 2 Mb, pero esto no es correcto.
El impacto en la performance según los diferentes tamaños del stripe es más difícil de cuantificar que con el ancho:

- **Disminuir el tamaño del stripe:** Al disminuir el tamaño del stripe se hace que los archivos se dividan en partes más pequeñas y utilicen más discos, disminuyendo así el tiempo de transferencia pero aumentando el de posicionamiento.
- **Aumentar el tamaño del stripe:** Permite que los archivos entren en menos bloques, disminuyendo el tiempo de posicionamiento y aumentando el de transferencia. El tiempo de posicionamiento se ve mejorado en caso que la controladora tenga la inteligencia suficiente para atender otro requerimiento en discos que no se utilicen dentro del array.

Como siempre se desprende de este pequeño análisis la pregunta a la que todo el mundo quisiera tener una respuesta tajante e indiscutible que es "*¿Cuál es el mejor tamaño para el stripe?*", y como siempre también la respuesta es "*Depende*". Hay innumerables recetas para esto, por ejemplo aquella que dice que se debería hacer coincidir el tamaño del stripe con el tamaño del cluster de un sistema de archivos FAT. Es una receta tan válida como inválida. Pese a que es casi imposible asegurarse que un cluster quede almacenado exactamente en un bloque, si lo fuese eso sólo mejoraría mi performance si lo único que me interesa es poder disminuir el tiempo de posicionamiento y no me interesa el de transferencia.
Por otro lado se podrían dictar recetas como que en sistemas transaccionales, donde hay una gran cantidad de lecturas y escrituras pequeñas, lo que conviene es utilizar un stripe size grande y en sistemas o aplicaciones que realizan pocas lecturas o escrituras de gran tamaño utilizar un stripe size pequeño.
No obstante, éstas son meras recetas y si uno realmente quiere saber qué stripe size es el mejor para su aplicación no queda otra alternativa que realizar un estudio con diferentes tamaños.
Lo que sí hay que saber al respecto, y esto no es de menor importancia, es que en la mayoría de los casos no se observarán grandes diferencias de performance con los distintos tamaños de stripe, excepto que se elijan tamaños en los extremos, es decir, muy grandes o muy pequeños.

Operación degradada y reconstrucción

Todos los análisis de performance de los diferentes niveles de RAID como de los diferentes parámetros configurables (ejemplo, tamaño del stripe) se basan en el funcionamiento normal del sistema.
Sin embargo todo cambia cuando un disco se rompe. En este caso el sistema pasa a lo que se denomina "Operación Degradada o en inglés, Degrade Operation". Se dice que la operación está degradada porque ahora el sistema tiene un disco menos (o dos según el nivel de RAID) y el hardware que queda

funcionando tiene que compensar esta pérdida. Por ejemplo, en un sistema RAID que funciona con mirroring, ahora el sistema ya no tiene dos discos sino sólo uno y su performance es la misma que cuando se tiene un solo disco. El caso es aún peor cuando el sistema utiliza striping con paridad, donde por cada operación de lectura el sistema tiene que "calcular" que datos son los que estaban en el disco o los discos que se perdieron y esto por supuesto, es pérdida de tiempo.
Por otro lado, un gran detractor de la perfomance en estos casos es la reconstrucción de datos (o rebuilding en inglés), que toma lugar cuando se reemplaza el disco fallado con uno nuevo. Esta reconstrucción, que puede llegar a tomar horas, es el proceso por el cual el sistema tiene que volver a "llenar" el disco nuevo de reemplazo con los datos que se encontraban en el que falló. Esta operatoria puede ser sencillamente copiar los datos del disco que quedó sano en el nuevo (en los sistemas que utilizar mirroring) como tan compleja como tener que calcular todas las paridades en el disco nuevo (en los sistemas que utilizan striping con paridad).
Los sistemas actuales de almacenamiento permiten que el administrador elija si esa reconstrucción se hará automáticamente, en caso que el sistema disponga de lo que se llama disco de *spare*, donde el administrador deja uno de todos los discos fuera de línea (off-line) y el mismo entra en juego cuando uno perteneciente a algún array en el sistema se rompe; o bien se hará manualmente, donde el administrador elige el momento en que se hará la reconstrucción. Debido a que la pérdida de performance es generalmente considerable, muchas veces no es mala idea esperar a horarios de baja carga transaccional para realizar el trabajo, no obstante teniendo en cuenta que en la mayoría de las configuraciones cuando un disco se rompe ya no queda lugar a la rotura de un segundo disco sin que esto lleve a la pérdida de información.

Fiabilidad, Tolerancia a Fallos, Disponibilidad y Recuperación de Datos

El concepto más utilizado para expresar la fiablidad de un componente o sistema es el *Mean Time Between Failures (MTBF) o tiempo medio entre fallos.* Lógicamente, un componente con un MTBF más alto es más fiable que uno con uno menor.
Matemáticamente hablando, la fiabilidad de todo un sistema viene dado por la siguiente fórmula:

$$MTBFs = \frac{1}{\frac{1}{MTBF_1} + \frac{1}{MTBF_2} + \cdots + \frac{1}{MTBF_n}}$$

Donde:

$MTBF_S = Fiabilidad\ del\ sistema$
$MTBF_1 = Fiabilidad\ del\ componente\ 1$
$MTBF_n = Fiabilidad\ del\ componente\ n$

Si el tiempo medio entre falla de todos los componentes es igual, la fórmula se simplifica a:

$$MTBF_S = \frac{MTBF_n}{N}$$

Donde N es la cantidad de componentes.

Esto implica nada menos que la fiabilidad del sistema decae a medida que se agregan más componentes al mismo.
De las fórmulas anteriores, sobre todo de la primera se desprende un concepto que no está a la vista, por lo menos no al primer vistazo, y que no es menor. ¿Recuerdan ese dicho que dice que una cadena se rompe por el eslabón más débil? Acá ocurre algo similar. No importa cuán fiables sean los diferentes componentes que conforman el sistema, la fiabilidad de todo el sistema siempre será menor a la fiabilidad que tiene el componente de menor fiabilidad.
¿Con números para que quede más claro el concepto? Supongamos tenemos un sistema con cinco componentes cuyo MTBF es de 1000000 (un millón) de horas. La fiabilidad de todo el sistema es de 200000 horas (aplicando la segunda fórmula). Si intercambiamos uno de esos componentes por uno que tiene un MTBF de 50000 horas, la confiabilidad de todo el sistema será tan solo de 41666 horas, casi una quinta parte del primero por sólo haber cambiado uno de los componentes.

Por lo tanto, ¿por qué entonces se dice que un sistema RAID es más *seguro, confiable o fiable*, que un sistema sin RAID? ¿Acaso al tener un RAID no tengo más discos y hardware (controladora por ejemplo) y por lo tanto la fiabilidad disminuye? Sí, pero es aquí donde entra el concepto de tolerancia a fallos, donde si bien la fiabilidad de todo el sistema disminuye en comparación al tener un solo disco, la utilización de técnicas como la de mirroring o paridad hace que el sistema sea tolerante a esos fallos. Esa tolerancia a fallos depende de qué nivel de RAID tengamos implementado.
¿Y si utilizo un RAID0, donde existe striping pero no paridad? Bueno, en ese caso entonces se está disminuyendo la fiabilidad y además no tengo tolerancia a fallos. En definitiva, tengo un sistema menos seguro que si no tuviera el RAID.
La disponibilidad es otro concepto difícil de definir con palabras pero fácil de entender. Basta con dirigirse al diccionario de la Real Academia Española para encontrarse con la siguiente definición:

> ***Disponible:*** *Dicho de una cosa: Que se puede disponer libremente de ella…*

Es como que el concepto se sobreentiende pero es algo vaga la definición. Por suerte para nosotros este problema lingüístico no nos trae demasiados problemas y podríamos decir que básicamente la disponibilidad significa cuán listos están los datos para ser usados, o siguiendo un poco la línea de definiciones de la RAE podríamos decir cuán disponibles están los datos para ser utilizados.
Lo importante a entender sobre disponibilidad es que la misma depende de varios factores a saber:

- *Fiabilidad del hardware:* Como es lógico, cuanto más fiable sea el hardware que utilicemos, mayor disponibilidad tendrán nuestros datos. Cuando hablamos del hardware en un sistema de RAID no sólo nos referimos a los discos en sí mismos sino a todo el hardware que se encuentra involucrado en el sistema (controladoras, fuentes de alimentación, cables, etc.)
- *Tolerancia a fallos:* Cuánto más tolerante a fallos sea nuestro sistema de RAID más alta disponibilidad el mismo proveerá (por ejemplo, RAID1 permite que un disco falle y los datos todavía sean accesibles, RAID0 esto no lo permite, etc.). Otra vez, es importante que todo el hardware involucrado sea tolerante a fallos y no sólo los discos. Volviendo al dicho de la cadena y el eslabón más débil, de nada nos sirve tener los mejores discos del mercado, doble controladora, todo redundante y no tener por ejemplo una UPS que haga frente a los problemas eléctricos. Fíjense cómo un sistema que aparentemente tiene una tolerancia a fallos altísima, termina dependiendo del suministro de energía de una empresa eléctrica.
- *Hot swapping:* Esta facilidad nos permite intercambiar los diferentes componentes de hardware sin la necesidad de tener que “apagar” el sistema para efectuar las tareas de mantenimiento. Por lo tanto, un sistema que soporta *hot swapping* provee una mayor disponibilidad que otro que no lo soporta.
- *Reconstrucción automática:* Si el sistema tiene que estar en ejecución constantemente, los sistemas que soportan la reconstrucción automática, a través del uso de los discos de *spare* proveen mayor disponibilidad y esta facilidad es esencial.
- *Servicio:* No importa cuán tolerante a fallos diseñemos nuestro sistema, el hecho es que los sistemas alguna vez fallan. Y cuando esto ocurre, la disponibilidad, por más que ya no sea del 100% depende de cuán rápido sea el servicio para poner nuevamente el sistema en producción.

Por último, aunque se cae de maduro, cuánto más alta queramos que sea la disponibilidad, más costoso será nuestro sistema. Sin embargo, hay muchísimos negocios en los que tener el sistema fuera de servicio un par de horas representa una pérdida mucho mayor al costo de la solución.
Llegados a este punto, otra vez es importante mencionar lo importante que son los backups. Si bien los sistemas con RAID proveen mayor disponibilidad (por lo menos así debería ser, excepto un mal diseño), los problemas ocurren y esto es un hecho. Y frente a la pérdida de información, es muy importante saber que la recuperación de datos es muchísimo más difícil de llevar a cabo en un sistema RAID que en un sistema con un solo disco. Piensen solamente en un sistema con RAID0, donde los datos están esparcidos por todos los discos y la mayoría de los archivos están divididos en varios discos físicos. Si un solo disco quedase fuera de servicio la tarea de reconstruir los datos sería una tarea realmente muy difícil.
La siguiente lista son fallas que un sistema RAID no soluciona:

- *Fallas inesperadas del hardware:* por ejemplo, que se rompan dos discos al mismo tiempo (pasa!!!)
- *Fallas en el hardware asociado al sistema RAID además de los discos*
- *Daños físicos:* Por ejemplo, un incendio en el datacenter (otra vez, pasa!!!)
- *Problemas de software:* Un bug de software que hace que se pierda la información
- *Virus, gusanos, etc.*
- Error humano: Casi, si no la mayor, causa de la pérdida de información (en el mundo de networking se diría *error de capa 8*)

Niveles de RAID

Finalmente llegamos al apartado en donde explicaremos los diferentes niveles de RAID. Como hablaremos en este apartado de las características de cada uno de los niveles, hablaremos primero de cada uno de los atributos que luego serán nombrados en cada uno de los diferentes niveles para evitar duplicación de datos.

Técnica de redundancia: Mirroring Vs. Paridad

La principal diferencia entre los niveles de RAID es la técnica utilizada para ofrecer redundancia, la cual se puede lograr mediante la técnica de *mirroring* o mediante la técnica de *paridad* (la cual se implemente siempre a su vez con *striping*).

- *Mirroring*: Utilizan esta técnica el nivel simple 1 de RAID y los niveles múltiples 0+1 y 1+0 (también conocido como RAID10). Hay una variante en que también se realiza un *mirroring* de la controladora RAID llamada *duplexing*. En realidad, en el mundo de los *storage*, esto se conoce como solución de storage con controladora redundante o *dual controller*.
- *Striping con paridad*: Utilizan esta técnica los niveles simples 2 al 7 de RAID y los niveles múltiples 0+3, 3+0, 0+5 y 5+0.
- *Ni mirroring ni paridad*: Esta técnica es utilizado por el nivel simple 0 de RAID, donde no se provee redundancia.
- *Tanto mirroring como striping con paridad*: Sólo se utiliza esta técnica en los niveles múltiples de RAID, como son 1+5 y 5+1.

La forma en que se realiza el striping justamente varía de nivel en nivel, donde algunos realizan striping a nivel de byte o sectores y otros sobre bloques más grandes, y como vimos, en estos últimos se puede configurar el *stripe size*. Particularmente en el nivel 2 de RAID se utiliza una técnica de paridad donde la misma no se calcula como vimos en uno de los apartados anteriores sino de una manera mucho más parecido al método ECC, utilizado por ejemplo en las memorias.

Capacidades de los array y eficiencia de almacenamiento

Una cosa muy importante acerca de los arrays de discos que trabajan en RAID, es que generalmente se necesitan discos dentro del array de iguales características y tamaño. Si bien muchas de las controladoras de discos en estos momentos soportan discos de diferente tamaño, siempre hay un desaprovechamiento en la capacidad. Por ejemplo, si se tuvieran dos discos, uno de 146 Gb y otro de 300 Gb y se los configurara para trabajar en RAID1 (mirroring), la capacidad del array sería de sólo 146 Gb, desperdiciándose gran parte de la capacidad del disco más grande. Hay que decir que algunas controladoras permiten utilizar este espacio desperdiciado en los discos de mayor capacidad como espacio común, es decir, espacio sin RAID.
Cuando hablamos de la capacidad total de un RAID aparece ligado el concepto de *Eficiencia de Almacenamiento* o en inglés, *Storage Efficiency*. El mismo se define como la capacidad utilizable del RAID dividido la suma de las capacidades de cada uno de los discos físicos que conforman dicho RAID. Éste es un valor que en algunos niveles de RAID se mantiene constante (por ejemplo, en RAID0 es el 100%, en RAID1 el 50%) mientras que en otros niveles varía según la cantidad de discos en el RAID (por ejemplo, en un RAID5 con 3 discos sería de 66%, con 4 discos del 75%). Si volvemos un párrafo hacia atrás donde comentábamos que cuando se utilizan discos de diferente tamaño aparece espacio desperdiciado en los discos de mayor tamaño, esto hace lógicamente que disminuya esta *eficiencia de*

almacenamiento, ya que las capacidades utilizables de los RAID se calculan en base al disco de menor tamaño del conjunto que forman el RAID. Por ejemplo, tres discos de 20, 40 y 60 GB configurados en nivel RAID0 hará que la capacidad del mismo sea de 60 Gb, donde la *eficiencia de almacenamiento* en este caso sería tan sólo del 50% versus el 100% que se obtiene al utilizar discos de igual tamaño.

Operación degradada y reconstrucción

Si bien ya dedicamos un apartado a estos dos conceptos, cuando hablamos de los mismos en relación al nivel de RAID utilizado lo más importante a destacar es cuán degradada se ve la performance cuando se pierde un discos del array y cuán costosa es su reconstrucción. Lo más llamativo de esto es que en muchos casos dichos parámetros se ven modificados sustancialmente al intercambiar la manera en que se construyen los niveles múltiples de RAID, por ejemplo RAID 5+0 ó RAID 0+5. ¿Qué quiero decir con esto? Cuando veamos los niveles múltiples de RAID más adelante, observaremos valores muy diferentes en estos parámetros si el RAID se arma de una u otra manera.

Performance

La performance o rendimiento no sólo varía entre los diferentes niveles de RAID (de hecho, muchísimas veces la elección de un determinado nivel de RAID se escoge basándose en este parámetro) sino que además diferentes tipos de performance lo hacen. Analizaremos en este resumen la performance asociadas a las lecturas y escrituras, como así también al posicionamiento (accesos aleatorios) y la transferencia (accesos secuenciales). Definiremos por tanto una matriz con estas cuatro variables:

- **Performance de lecturas aleatorias:** Estudiaremos cómo se comporta determinado nivel de RAID en lecturas aleatorias. Muy importante para sistemas transaccionales donde el ratio de lectura Vs. escritura es elevado y los archivos a leer son pequeños. Un ejemplo de este tipo de sistemas es un portal de compras web, donde la mayoría de las consultas a la base de datos son lecturas, y sólo se producen escrituras cuando por ejemplo un usuario publica un artículo, o realiza una oferta, etc. En este caso, la diferencia entre lecturas y escrituras es muy significativa.
- **Performance de escrituras aleatorias:** Cómo se comporta determinado nivel de RAID frente a las escrituras aleatorias. De extrema importancia en sistemas transaccionales donde se realizan muchas escrituras ya que generalmente, la performance asociada a las escrituras en los diferentes niveles de RAID es mucho peor que la de lectura. Podemos pensar por ejemplo en algún sistema que realiza auditorías donde la mayoría de las operaciones son de escrituras, ya que básicamente lo único que hacen es grabar información en la base de datos de acciones que realizan los diferentes agentes, y cuando es necesario, se consultan dichas acciones (casos en que se accedería para leer).
- **Performance de lecturas secuenciales:** Cómo se comporta determinado nivel de RAID al leer grandes archivos secuenciales. Por supuesto muy importante en sistemas donde la relación entre lecturas y escrituras es elevada, como por ejemplo un servidor de video.
- **Performance de escrituras secuenciales:** La performance de un determinado nivel de RAID al escribir archivos de gran tamaño, por ejemplo, sistemas de edición de video.

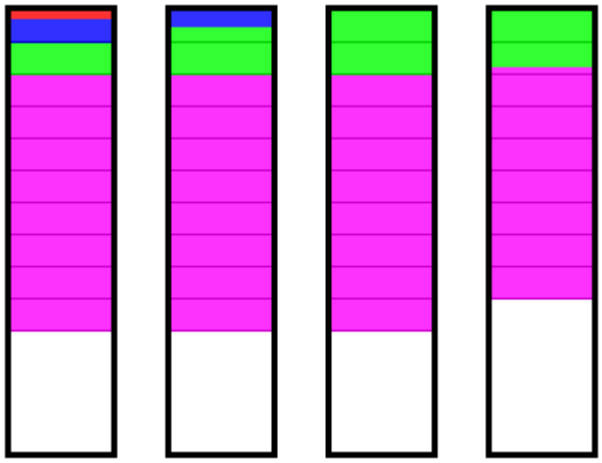
Niveles de RAID simples

Estos niveles de RAID son los más ampliamente utilizados, en contraposición de los niveles de RAID múltiples que veremos más adelante. Salvo raras excepciones de requerimientos críticos (tanto de performance como de tolerancia a fallos), estos niveles de RAID satisfacen la mayoría de los requerimientos cotidianos.
Son ocho niveles, donde tres de ellos, el 0, el 1 y el 5 son los más populares, mientras que algunos son extremadamente difíciles de ver hoy en día.

RAID 0

- **Nombre con el que se lo conoce:** RAID0
- **Técnica utilizada:** Striping (sin paridad)

- **Descripción:** Es el nivel más simple de todos los RAID. Los archivos son partidos en bloques (stripes) de tamaño definido por el usuario y cada stripe se envía a un disco diferente dentro del array. Debido a que no posee redundancia, es el nivel de RAID que mejor performance en general presenta.



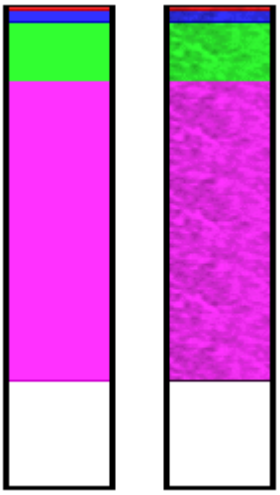
En el ejemplo se muestra un array de cuatro discos con un stripe size de 16 Kb. El archivo de color rojo ocupa 4 Kb, el azul 20 Kb, el verde 100 Kb y el magenta 500 Kb. Cada pixel vertical representa 1 Kb.

- **Requerimientos de la controladora:** Soportado por todas las controladoras de hardware y software
- **Requerimientos de discos:** Mínimo 2 discos. Máximo según la controladora. Se deberían utilizar de iguales características y tamaño para eliminar el desperdicio
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos en el array
- **Eficiencia de almacenamiento:** 100% si se utilizan todos los discos de igual tamaño
- **Tolerancia a fallos:** Ninguna.
- **Disponibilidad de datos:** La más baja de todos los niveles de RAID. La no tolerancia a fallos hace que la recuperación de datos sea muy difícil de llevar a cabo
- **Degradación y reconstrucción:** No aplica
- **Performance de lecturas aleatorias:** Muy buena. Aún mejor si se utiliza un tamaño de stripe grande y la controladora permite realizar lecturas independientes en diferentes discos del array
- **Performance de escrituras aleatorias:** Muy buena. Ídem anterior pero para las escrituras
- **Performance de lecturas secuenciales:** Muy buena a excelente
- **Performance de escrituras secuenciales:** Muy buena
- **Costo:** El más bajo de todos los niveles
- **Consideraciones especiales:** Realizar backup periódico de la información en este tipo de RAID
- **Usos recomendados:** Datos no críticos (o que no cambian muy seguido y del cual se realiza backup periódico) donde se necesita gran velocidad. Audio y video streaming, servidores web, diseño gráfico

RAID 1

- **Nombre con el que se lo conoce:** RAID1, RAID1 con duplexing
- **Técnica utilizada:** Mirroring o duplexing

Descripción: Este nivel de RAID generalmente se encuentra implementado con mirroring. Un disco tiene todos sus datos duplicados en dos discos. Se puede realizar por hardware o por software. Si alguno de los dos discos falla el otro contiene exactamente la misma información. Una variante de este nivel es el implementado con duplexing, donde además de duplicar el disco se duplica la controladora.



En este ejemplo se ve cómo los datos están duplicados en los dos discos. Los archivos son los mismos que para el ejemplo de RAID0.

- **Requerimientos de la controladora:** Soportado por todas las controladoras de hardware y software
- **Requerimientos de discos:** Exactamente 2 discos. Se deberían utilizar de iguales características y tamaño para eliminar el desperdicio
- **Capacidad del array:** Tamaño del disco de menor capacidad
- **Eficiencia de almacenamiento:** 50% si se utilizan todos los discos de igual tamaño. Si no tamaño del menor disco dividido la suma de los tamaños de ambos
- **Tolerancia a fallos:** Muy buena. Si se utiliza duplexing aún mejor.
- **Disponibilidad de datos:** Muy buena. La mayoría de las controladoras soporta los discos de spare y al auto reconstrucción del array
- **Degradación y reconstrucción:** Ligera degradación de performance (frente a un fallo, claro está) para las lecturas. Para las escrituras la performance mejora levemente.
- **Performance de lecturas aleatorias:** Buena. Mejor que para un disco solo pero peor que para muchos otros niveles
- **Performance de escrituras aleatorias:** Buena. Peor que para un disco solo pero mejor que para muchos otros niveles
- **Performance de lecturas secuenciales:** Razonable, casi la misma que para un disco solo
- **Performance de escrituras secuenciales:** Buena, mejor que para muchos otros niveles
- **Costo:** Relativamente alto debido al disco redundante y la baja eficiencia de almacenamiento. Más caro aún si se utiliza duplexing, donde también existe una controladora redundante. Sin embargo, el tipo de controladora a utilizar es muy sencilla y de bajo costo comparado con otras más complejas
- **Consideraciones especiales:** Está limitado en tamaño al tamaño de los discos. Se pueden utilizar más de un RAID 1 si se necesita más capacidad, pero en estos casos comienza a ser más atractivo el uso de RAID 1+0
- **Usos recomendados:** Aplicaciones que requieren tolerancia a fallos a un costo razonable, especialmente en los casos en que no se necesita gran capacidad almacenamiento

RAID 2

- **Nombre con el que se lo conoce:** RAID 2
- **Técnica utilizada:** Striping a nivel de bit con código de Hamming ECC
- **Descripción:** Es el único nivel de RAID que utiliza una técnica diferente de paridad a la utilizada en los niveles 3 al 7. Realiza un striping de los datos a nivel de bits y los aloja en diferentes discos de datos y de paridad. Los bits redundantes se calculan utilizando el código de Hamming de corrección de errores (ECC). Cada vez que se tiene que escribir algún dato se escriben los bits pertenecientes al mismo, se calculan los bits de paridad y se los escribe en los discos de paridad.

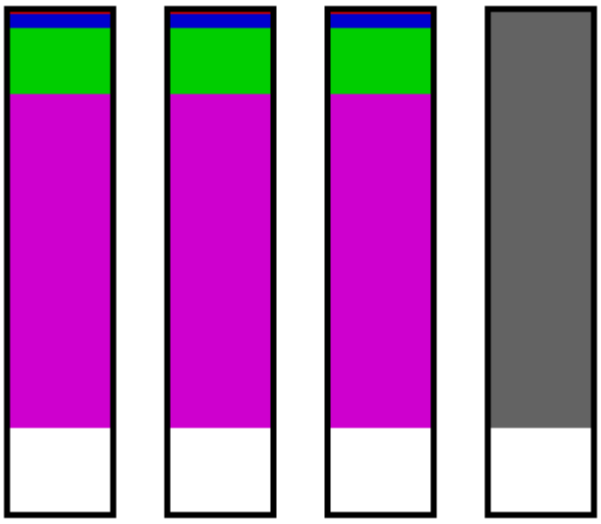
Cuando se leen los datos, se leen los bits desde los discos de datos y también se leen los bits de paridad, entonces se calcula nuevamente la paridad de los datos leídos y se los compara a los leídos directamente de los discos de paridad para asegurarse que los datos no fueron modificados desde su escritura. Si se detectó algún error, el mismo puede ser corregido "on the fly". Es el mismo concepto que utilizan los discos de hoy en día internamente para verificar que los datos son correctos.
Es el único nivel de RAID definido originalmente que hoy en día ya no se utiliza para nada, debido a la restricción en la cantidad de discos a utilizar y a que las controladoras requeridas eran muy complejas y costosas. La performance también era baja con respecto a los demás niveles, pero sobre todo, se dejó de utilizar ya que el control de paridad utilizado para la detección de errores hoy en día ya está implícito en la electrónica de todos los discos.
Debido a su escasez en el mercado, los datos a continuación son más provenientes de un análisis teórico que de la práctica.

- **Requerimientos de la controladora:** Controladora por hardware específica
- **Requerimientos de discos:** Según la implementación, pero típicamente 10 discos de datos más cuatro de paridad, lo que hacen un total de 14. Ó bien 32 discos de datos y 7 de paridad, total 39. Los discos eran sincronizados en su giro para poder trabajar en tándem.
- **Capacidad del array:** Depende de la implementación, pero con los discos de hoy en día sería muy grande.
- **Eficiencia de almacenamiento:** También depende de la implementación. Para la configuración de 10+4 de un 71%. Para la de 32+7 de un 81%.
- **Tolerancia a fallos:** Solamente razonable, ya que no provee mucha redundancia. Sólo puede fallar un disco para recuperar el dato "on the fly".
- **Disponibilidad de datos:** Muy buena debido a la corrección de errores "on the fly".
- **Degradación y reconstrucción:** En teoría sería muy poca la degradación frente al fallo de un disco.
- **Performance de lecturas aleatorias:** Razonable. El striping a nivel de bits hace imposible las lecturas simultáneas en varios discos a la vez.
- **Performance de escrituras aleatorias:** Mala debido al striping a nivel de bits y el cálculo de paridad.
- **Performance de lecturas secuenciales:** Muy buena debido al paralelismo de varios discos.
- **Performance de escrituras secuenciales:** Razonable a buena.
- **Costo:** Muy elevado.
- **Consideraciones especiales:** No se utiliza en los sistemas de hoy en día.
- **Usos recomendados:** No se utiliza en los sistemas de hoy en día.

RAID 3

- **Nombre con el que se lo conoce:** RAID 3 (cuidado que muchos fabricantes dicen implementar RAID 3 cuando realmente están implementado RAID 4)
- **Técnica utilizada:** Striping a nivel de byte con paridad dedicada.

Descripción: En este nivel de RAID los datos son divididos en múltiples discos a nivel de bytes. El número exacto de bytes enviados a cada disco (el tamaño del stripe) varía pero típicamente es 1024. La información de paridad se envía a un disco dedicado a tal fin. Por ello, dicho disco generalmente se transforma en un cuello de botella en las escrituras, especialmente en las aleatorias, ya que cada vez que se tiene que escribir algo en el array se tiene que acceder a dicho disco. Los RAID 3 y 4 difieren sólo en el tamaño de stripe enviado a varios discos.



En este ejemplo se ve cómo los diferentes archivos son divididos entre los diferentes discos. Los archivos tienen el mismo tamaño que los ejemplos anteriores para los otros niveles de RAID ya vistos. Fíjense sin embargo que los datos de los archivos sólo son enviados a tres de los discos cuando el último se reserva sólo para la información relacionada con la paridad. Debido a que el tamaño de los stripes es tan pequeño, no se ve en el dibujo los límites entre bloque y bloque.

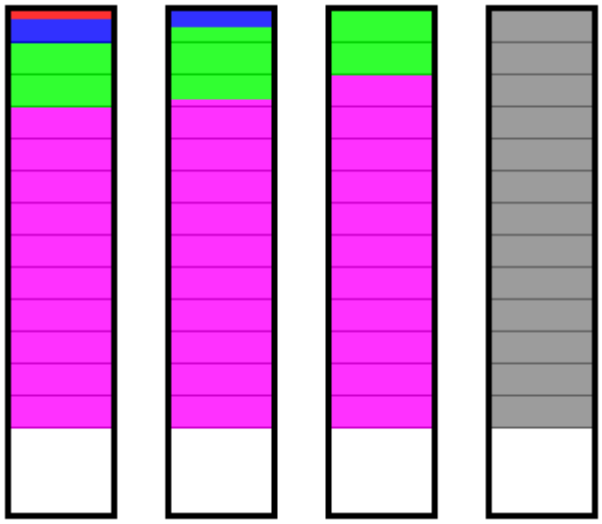
- **Requerimientos de la controladora:** Generalmente se requiere de una controladora por hardware de mediana a alta gama (por ende, cara)
- **Requerimientos de discos:** Mínimo 3 discos. Máximo según la controladora. Debieran ser del mismo tamaño.
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos menos 1.
- **Eficiencia de almacenamiento:** Si todos los discos son del mismo tamaño, número de discos en el array menos uno, dividido el número de discos.
- **Tolerancia a fallos:** Buena. Puede tolerar la pérdida de un disco.
- **Disponibilidad de datos:** Muy buena. La mayoría de las controladoras soporta los discos de spare y al auto reconstrucción del array.
- **Degradación y reconstrucción:** Relativamente una pequeña degradación frente a un fallo. La reconstrucción puede tomar varias horas.
- **Performance de lecturas aleatorias:** Buena pero muy buena debido al striping a nivel de bytes.
- **Performance de escrituras aleatorias:** Mala debido al striping a nivel de byte, el cálculo de paridad y el cuello de botella producido en el disco de paridad.
- **Performance de lecturas secuenciales:** Muy buena.
- **Performance de escrituras secuenciales:** Razonable a buena.
- **Costo:** Moderado.
- **Consideraciones especiales:** No es tan popular como otros niveles. Para sistemas transaccionales el nivel RAID 5 demostró mejores características.
- **Usos recomendados:** Aplicaciones que trabajan con archivos muy grandes que requieren una gran transferencia de datos con redundancia.

RAID 4

- **Nombre con el que se lo conoce:** RAID 4 (muchas veces confundido con RAID 3)
- **Técnica utilizada:** Striping a nivel de bloque con paridad dedicada.

Descripción: Este nivel de RAID mejora la performance de RAID 3 realizando un striping a nivel de bloque en lugar de bytes y provee redundancia con un disco de paridad dedicado. Es como RAID 3 excepto que usa bloques en lugar de bytes para el striping y como el RAID 5 excepto que utiliza un disco dedicado a la paridad en lugar de distribuirla. Haciendo que el striping sea a nivel de bloques gana performance en los

accesos aleatorios con respecto al RAID 3 pero su paridad en un disco dedicado sigue siendo un cuello de botella.



En este ejemplo se ve cómo los diferentes archivos (de igual tamaño que en los casos anteriores) se distribuyen entre los diferentes discos, dejando a uno en especial para que aloje la información de paridad. La diferencia entre esta ilustración y la de RAID 3 es el tamaño de los bloques de cada stripe.

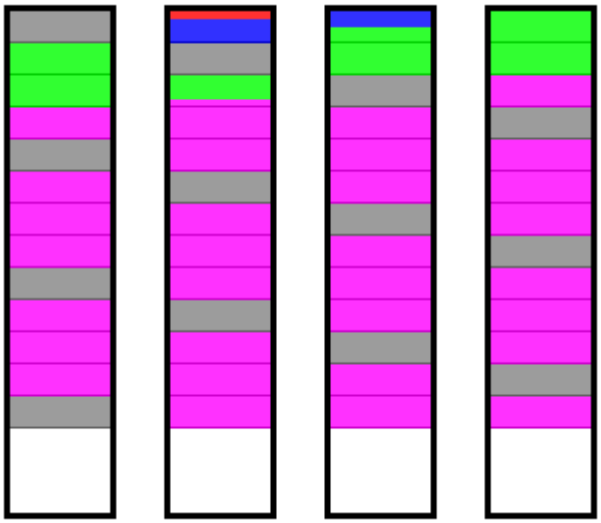
- **Requerimientos de la controladora:** Generalmente se requiere de una controladora por hardware de mediana a alta gama (por ende, cara)
- **Requerimientos de discos:** Mínimo 3 discos. Máximo según la controladora. Debieran ser del mismo tamaño.
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos menos 1.
- **Eficiencia de almacenamiento:** Si todos los discos son del mismo tamaño, número de discos en el array menos uno, dividido el número de discos.
- **Tolerancia a fallos:** Buena. Puede tolerar la pérdida de un disco.
- **Disponibilidad de datos:** Muy buena. La mayoría de las controladoras soporta los discos de spare y al auto reconstrucción del array.
- **Degradación y reconstrucción:** Moderada degradación frente a un fallo. La reconstrucción puede tomar varias horas.
- **Performance de lecturas aleatorias:** Muy buena.
- **Performance de escrituras aleatorias:** Mala a razonable debido al cálculo de paridad y el cuello de botella producido en el disco de paridad.
- **Performance de lecturas secuenciales:** Buena a muy buena.
- **Performance de escrituras secuenciales:** Razonable a buena.
- **Costo:** Moderado.
- **Consideraciones especiales:** La performance dependerá en cierta medida al tamaño del stripe utilizado.
- **Usos recomendados:** No está casi utilizado hoy en día, ya que el RAID 5 aplica en todos los casos que podría aplicar el RAID 4 presentando mejor performance.

RAID 5

- **Nombre con el que se lo conoce:** RAID 5
- **Técnica utilizada:** Striping a nivel de bloque con paridad distribuida.

Descripción: Es uno de los niveles de RAID más populares de hoy en día. Es en definitiva similar al RAID 4 pero también distribuye entre todos los discos del array a la paridad. De esta manera desaparece el cuello de botella que se generaba en RAID 4 y mejora sensiblemente la performance en las escrituras

aleatorias, además de permitir un mejoramiento en el paralelismo en ambientes multi transaccionales. Por supuesto, el cálculo y la escritura de paridad por cada acceso de escritura hacen que la performance no sea tan buena para la misma. La tolerancia a fallos se logra asegurándose que la paridad para un determinado bloque se distribuya en diferentes discos separados a la información en sí. La performance de este nivel de RAID se puede "ajustar" intentando configurar diferentes tamaños de stripe según la aplicación a ejecutar en el sistema.



En este ejemplo se ve cómo los diferentes archivos (de igual tamaño que en los casos anteriores) se distribuyen entre los diferentes discos pero a diferencia del caso anterior, también está distribuida la información de paridad entre los mismos.

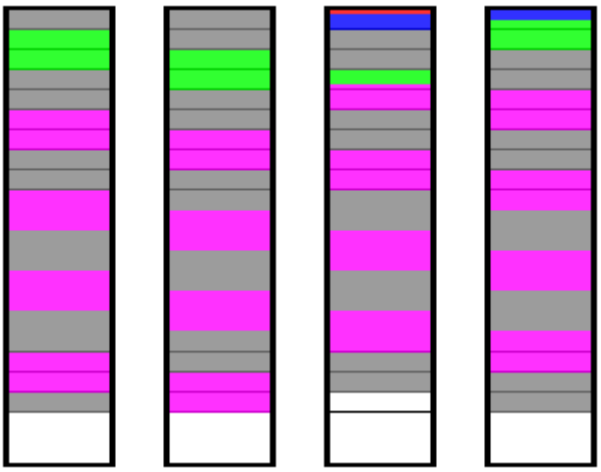
- **Requerimientos de la controladora:** Generalmente se requiere de una controladora por hardware de alta gama. Algunos sistemas operativos permiten realizar una implementación por software, pero a un costo muy elevado de pérdida de performance.
- **Requerimientos de discos:** Mínimo 3 discos. Máximo según la controladora. Debieran ser del mismo tamaño.
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos menos 1.
- **Eficiencia de almacenamiento:** Si todos los discos son del mismo tamaño, número de discos en el array menos uno, dividido el número de discos.
- **Tolerancia a fallos:** Buena. Puede tolerar la pérdida de un disco.
- **Disponibilidad de datos:** Buena a muy buena. La mayoría de las controladoras soporta los discos de spare y al auto reconstrucción del array (las implementaciones por software requieren de down-time)
- **Degradación y reconstrucción:** Debido a la distribución de la paridad, la degradación luego de un fallo puede ser sustancial, al igual que la reconstrucción.
- **Performance de lecturas aleatorias:** Muy buena a excelente, generalmente mejor aún si el tamaño del stripe es grande.
- **Performance de escrituras aleatorias:** Sólo razonable debido al cálculo de paridad. Si bien mejor que en RAID 3 y 4 debido a la eliminación del disco de paridad dedicado.
- **Performance de lecturas secuenciales:** Buena a muy buena, generalmente mejor aún para tamaños de stripe pequeños.
- **Performance de escrituras secuenciales:** Razonable a buena.
- **Costo:** Moderado, pero menor a RAID 3 y 4 debido a su alta popularidad. Y menor aún si se lo implemente por software.
- **Consideraciones especiales:** La performance dependerá en cierta medida al tamaño del stripe utilizado.
- **Usos recomendados:** Este nivel de RAID es visto por muchos como el ideal por su combinación de buena performance, alta capacidad, y alta eficiencia de almacenamiento. Generalmente es

utilizado para sistemas de propósito general. Para aplicaciones de alta demanda de escrituras, los niveles RAID 1 y RAID 1+0 son una mejor opción.

RAID 6

- **Nombre con el que se lo conoce:** RAID 6, aunque algunos fabricantes realizan implementaciones propietarias basadas en RAID 5 y a éstas, las llaman RAID 6.
- **Técnica utilizada:** Striping a nivel de bloque con paridad doble distribuida.

Descripción: Es un nivel de RAID muy parecido al RAID 5 sólo que calcula dos conjuntos de paridad, permitiendo así una tolerancia a fallos mayor que los demás niveles simples de RAID, ya que permite que fallen hasta dos discos. Al realizarse los cálculos de dos conjuntos de paridad es de entender que la performance en la escritura sea un poco menos a la de RAID 5. Sin embargo, al utilizarse un disco más la performance en las lecturas aleatorias es sensiblemente mayor. Como siempre, dicha performance puede variar al experimentar con diferentes tamaños del stripe.



En este ejemplo se ve cómo los diferentes archivos (de igual tamaño que en los casos anteriores) se distribuyen entre los diferentes discos pero a diferencia del caso anterior la paridad ocupa el doble de espacio en dos discos por separado.

- **Requerimientos de la controladora:** Se requiere de una controladora de alta gama (por ende, de alto costo)
- **Requerimientos de discos:** Mínimo 4 discos. Máximo según la controladora. Debieran ser del mismo tamaño.
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos menos 2.
- **Eficiencia de almacenamiento:** Si todos los discos son del mismo tamaño, número de discos en el array menos dos, dividido el número de discos.
- **Tolerancia a fallos:** Muy buena a excelente. Puede tolerar la pérdida de dos discos cualesquiera.
- **Disponibilidad de datos:** Excelente.
- **Degradación y reconstrucción:** Debido a la existencia de dos conjuntos de paridad y a la complejidad del manejo de las mismas, la pérdida de performance en el estado degradado puede ser importante. Generalmente se planifica la reconstrucción para horas no pico para no impactar demasiado la performance durante este proceso.
- **Performance de lecturas aleatorias:** Muy buena a excelente, generalmente mejor aún si el tamaño del stripe es grande.
- **Performance de escrituras aleatorias:** Mala, debido a la doble paridad y complejidad de algoritmos.
- **Performance de lecturas secuenciales:** Buena a muy buena, generalmente mejor aún para tamaños de stripe pequeños.
- **Performance de escrituras secuenciales:** Razonable.

- **Costo:** Alto.
- **Consideraciones especiales:** No es muy utilizado.
- **Usos recomendados:** Cuando se lo utiliza, se hace en los mismos ambientes en que se utiliza RAID 5 pero con una necesidad muy alta de tolerancia a fallos. Muchas empresas no están dispuestas a invertir el costo que una implementación RAID 6 implica, ya que no sólo requiere de una controladora capaz de soportar este nivel, sino que además se pierden dos discos de capacidad.

RAID 7

- **Nombre con el que se lo conoce:** RAID 7
- **Técnica utilizada:** Striping con cache asincrónico y con paridad dedicada.
- **Descripción:** A diferencia de los demás niveles de RAID, este nivel no es estándar. De hecho es una marca registrada de la firma Storage Computer Corporation. Debido a esto, la información que hay al respecto es limitada. Sin embargo se sabe que trabaja de manera similar a los niveles de RAID 3 y 4, pero de manera mucho más optimizada y mejorada para contrarrestar los efectos negativos de performance que tienen estos niveles. Básicamente logra esto a través del uso de memorias cache de varios niveles y un procesador de uso específico para manejar los discos de manera asincrónica. Claro está que estos excelentes niveles de performance se logran a base de una gran inversión en el hardware.
- **Requerimientos de la controladora:** Requiere de una controladora especial, de alto costo y propietaria.
- **Requerimientos de discos:** Depende de la implementación.
- **Capacidad del array:** Depende de la implementación.
- **Eficiencia de almacenamiento:** Depende de la implementación.
- **Tolerancia a fallos:** Muy buena.
- **Disponibilidad de datos:** Excelente, debido a la posibilidad de tener varios discos de reemplazo.
- **Degradación y reconstrucción:** Mejor que la mayoría de los niveles de RAID debido al hardware dedicado y al cache de múltiples niveles.
- **Performance de lecturas aleatorias:** Muy buena a excelente. Básicamente debido a que muchas lecturas las soluciona el cache sin tener la necesidad de acceder físicamente a los discos.
- **Performance de escrituras aleatorias:** Muy buena, generalmente mucho mejor que en los demás niveles que utilizan striping con paridad.
- **Performance de lecturas secuenciales:** Muy buena a excelente.
- **Performance de escrituras secuenciales:** Muy buena.
- **Costo:** Muy elevado.
- **Consideraciones especiales:** Es una solución propietaria no estándar. El uso del cache hace que ése sea un punto vulnerable en la solución, por eso se recomienda el uso de UPS. Sin embargo, las controladoras de discos de alta gama vienen ya equipadas con baterías que les brindan la energía necesaria para poder volcar todos los datos que contienen en memoria a los discos.
- **Usos recomendados:** Usos muy específicos en los que se necesitan altos niveles de performance. Para muchas situaciones, el uso de niveles múltiples de RAID como 1+0 es más que suficiente.

Niveles de RAID Múltiples o Anidados

Generalmente se utilizan los niveles múltiples de RAID porque presentan mejores niveles de performance que los dos niveles simples de RAID por separado por los que están compuestos.

RAID X+Y Vs. Y+X

La manera de obtener múltiples niveles de RAID es agarrar un conjunto de discos físicos y separarlos en grupos. Luego, dentro de cada grupo se aplica un nivel de RAID simple para formar un número de arrays. Luego, se aplica el segundo nivel simple de RAID a los arrays formados al principio. Por eso es que existen dos maneras de formar los niveles múltiples de RAID, ya que al haber dos niveles, importa cuál de ellos se aplica primero. Y no sólo eso, sino que muchas veces se obtienen características deferentes

según el orden en que se aplicaron. Por tanto es importante saber que el nivel de RAID 0+1 es diferente al RAID 1+0, o genéricamente hablando, el nivel de RAID X+Y es diferente al RAID Y+X.
Veremos a continuación un ejemplo para clarificar este concepto.
Supongamos un conjunto de 10 discos físicos en los que queremos aplicar los niveles de RAID simples 0 y 1. A estos 10 discos los podemos pensar como 2 conjuntos de 5 discos cada uno, o bien 5 conjuntos de 2 discos físicos cada uno. Por lo tanto podemos crear nuestros dos niveles de RAID múltiples posibles de la siguiente manera:

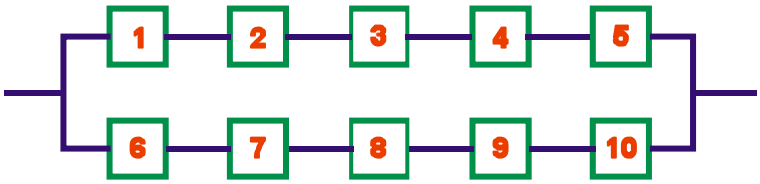
- **RAID 0, luego RAID 1 (RAID 0+1):** Se dividen los 10 discos físicos en dos grupos de 5 cada uno. A cada uno de los grupos le aplicamos el RAID 0, con lo que obtenemos en cada uno de los grupos 5 discos utilizando la técnica de striping sin paridad. Luego, aplicamos la técnica de mirroring (RAID 1) a los dos grupos. Como en definitiva lo que hacemos es un mirror luego de haber hecho el striping, este nivel de RAID se lo conoce como *"mirror of stripe"*
- **RAID 1, luego RAID 0 (RAID 1+0):** En este caso, dividimos a los 10 discos físicos en 5 grupos de 2 discos cada uno. A cada uno de los grupos le aplicamos la técnica de mirroring (RAID 1). Luego, aplicamos la técnica de striping (RAID 0) a cada uno de los 5 grupos que ya estaban espejados. Por eso la técnica utilizada en este nivel de RAID se conoce como "*stripe of mirrors"*

Por convención, cuando vemos un RAID X+Y significa que primero se aplica el nivel RAID X y luego el Y. Sin embargo, como siempre que se crea una convención, hay fabricantes que la rompen. Por eso hay quien llama RAID01 y RAID10 a lo mismo, otros que llaman RAID10 a lo que realmente es RAID01, etc. Por eso, es muy importante saber bien los conceptos de qué implica cada cosa para darse cuenta qué es lo que realmente un fabricante está implementando.
Por último, dedicaremos algunas líneas en este apartado a describir un problema que se presenta a menudo con muchas controladoras del mercado, y es referente a cómo varía la tolerancia a fallos dependiendo de qué nivel de RAID se aplique primero. Estas controladoras a las que hacemos referencia para implementar un nivel múltiple de RAID lo que hacen es crear un "super array" conformado por "sub arrays", consideran a estos sub arrays en un estado "en servicio" o "fuera de servicio" según si sus respectivos discos están bien o no, y hacen uso de la redundancia de datos dentro de los sub arrays pero no "entre" los sub arrays. Veámoslo con un ejemplo para que quede más claro. Para ello utilizaremos los mismos 10 discos de los ejemplos anteriores:

- **RAID 0+1:** En este caso, se utiliza la técnica de striping entre los discos 1, 2, 3, 4 y 5 (RAID 0 de cinco discos) para formar el sub array "A", y lo mismo con los discos 6, 7, 8, 9 y 10 para formar el sub array "B". Luego, realiza un mirroring entre los sub arrays "A" y "B". Ahora supongamos que el disco N° 2 de rompe. Como el sub array "A" está configurado como un RAID 0, y por lo tanto sin redundancia, todo este sub array quedará fuera de servicio. Ahora, si mientras este disco no es reemplazado sale de servicio el disco N° 9 perteneciente al segundo sub array, todo el sistema quedará fuera de servicio, ya que estas controladoras en cuestión no tienen la inteligencia para "darse cuenta" que la información del disco N° 9 es la misma que la del disco N° 4 si no se hubiera sacado de servicio al sub array "A" al fallar el primero disco y se hubiera seguido realizando el striping en los discos que quedaban activos. Esto hace que este nivel múltiple de RAID no permite que falle más de un disco si los mismos pertenecen a sub arrays diferentes.
- **RAID 1+0:** En este caso se forman ahora 5 sub arrays, formados por las duplas de discos 1 y 2; 3 y 4; 5 y 6; 7 y 8; y 9 y 10 utilizando la técnica de mirroring en cada una de ellas. Luego, se aplica el striping a todas las duplas. Ahora, si el disco 2 queda fuera de servicio el único sub array afectado será el N° 1 pero como todavía queda el disco 1 dentro el mismo en servicio, el sub array quedará activo. Suponiendo que en ese estado también falle el disco 9, como el mismo pertenece a otro sub array, también quedará en servicio este último y todo el sistema permanecerá activo. Por lo tanto, esta implementación con RAID 1+0 permite que queden fuera de servicio hasta 5 discos siempre y cuando los mismos pertenezcan a sub arrays diferentes.

Tal vez no quede tan claro, pero la segunda configuración (RAID 1+0) es más robusta que la primera (RAID 0+1). Para que no queden dudas al respecto, calcularemos la probabilidad de falla de cada una de las implementaciones.

Podemos pensar en la primer configuración como dos circuitos en paralelo (cada uno de los sub array) conformados cada uno por 5 componentes en serie (los 5 discos). Gráficamente se vería de la siguiente manera:



Con esto estamos indicando que si algún componente del "circuito" de arriba falla, todo ese circuito (sub array) fallará, y lo mismo con el de abajo. Los componentes serían nuestros discos físicos.
Ahora consideremos que los discos tienen una probabilidad de falla en un determinado tiempo igual a p_n donde *n* es el número de disco. Para simplificar los cálculos, sin dejar por ello de ser válido nuestro análisis, consideraremos que todos los discos son de idénticas características y por ende, las probabilidades de fallo en un determinado período de tiempo de cada uno de ellos es igual. Por lo tanto tenemos que $p_1 = p_2 = ... = P_n$.
Hablando en términos de probabilidades entonces, sabemos que *p* es la probabilidad de que un disco falle. Consideraremos que los sucesos son independientes, es decir, la probabilidad que el disco 1 falle es totalmente independiente de la probabilidad de que falle el disco n.
Para entender el siguiente desarrollo, tenemos que recordar que si *p* es la probabilidad de que un determinado evento suceda, entonces (1 – *p*) es la probabilidad de que dicho evento NO suceda. Por ejemplo, si tenemos un dado no cargado y la probabilidad de que salga el 1 es 1/6, la probabilidad de que el 1 NO salga será (1 – 1/6), es decir, 5/6.
Por lo tanto, si p_n es la probabilidad de que falle un determinado disco, entonces (1-p_n) será la probabilidad de que ese disco NO falle, o lo que es lo mismo, que funcione.
Hasta acá tenemos la probabilidad de que cualquiera de los discos funcione, siempre teniendo en cuenta el mismo período de tiempo. A continuación, lo que haremos será calcular la probabilidad conjunta de que los 5 discos que se encuentran en cada uno de los sub arrays funcionen. Para ello, multiplicamos cada una de las probabilidades de que funcione cada uno de los discos obteniendo:

$$P_{c5} = (1 - p_1).(1 - p_2) \dots (1 - p_5)$$

Donde P_{c5} es la probabilidad conjunta de que los 5 discos funcionen simultáneamente en el mismo período de tiempo. Pero como al principio dijimos que todas las probabilidades de fallo eran iguales, entonces la fórmula anterior se reduce a:

$$P_{c5} = (1 - p_n)^5$$

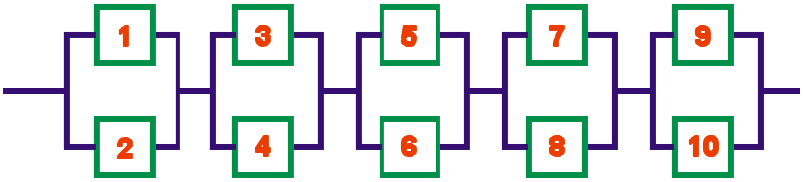
Ahora bien, P_{C5} como dijimos es la probabilidad de que todos los 5 discos funcionen al mismo tiempo. Si ahora hacemos:

$$P_{fn} = 1 - P_{c5} = 1 - (1 - p_n)^5$$

Obtenemos la probabilidad de que alguno de los 5 discos de cada uno de los sub arrays falle. P_{fn} es justamente esa probabilidad. Como P_{C5} era la probabilidad de que los 5 discos funcionen **al mismo tiempo**, 1 menos esa probabilidad nos da la probabilidad de que alguno falle.
Pero como son dos los sub arrays que hay y cada uno de ellos tiene 5 discos, para que el sistema total falle deben fallar los dos sub arrays simultáneamente. Por eso, la probabilidad de falla de todo el sistema está dada por:

$$P_T = [1 - (1 - p_n)^5]^2 \quad [1]$$

Para analizar el segundo caso, al igual que en el caso anterior, podemos considerarlo como un circuito serie de 5 circuitos paralelos (cada uno de los sub arrays) con 2 componentes cada uno (los discos). Gráficamente lo podríamos ver de la siguiente manera:



Ahora, en este caso, vemos que el sistema falla si falla cualquiera de los 5 sub arrays, pero no obstante, eso pasará sólo sí los dos componentes que forman el subarray fallan al mismo tiempo.
Buscamos obtener una fórmula que nos permita calcular la probabilidad de falla de este sistema dada la probabilidad de falla de cada uno de los componentes. Como en el análisis anterior, consideraremos que las probabilidades de falla de cada uno de los componentes son iguales y la llamaremos P_n.

Si P_n es la probabilidad de falla de cualquier componente, $(P_n)^2$ será igual a la probabilidad de que dos componentes fallen al mismo tiempo (en este caso cada uno de los dos discos que conforman cada sub array). Ahora bien, si $(P_n)^2$ es la probabilidad que dos componentes fallen al mismo tiempo, entonces:

$$(1 - {p_n}^2)$$

Será la probabilidad que alguno de esos dos componentes funcione. Ahora bien, lo que podemos hacer ahora es calcular la probabilidad conjunta que todos esos pares funcionen al mismo tiempo, obteniendo la siguiente fórmula:

$$(1 - {p_n}^2)^5$$

Por último, como la última fórmula nos indica la probabilidad que las 5 duplas de componentes funcionen al mismo tiempo (recordemos que una dupla se considera que funciona si al menos funciona uno de sus dos componentes), podemos calcular lo siguiente:

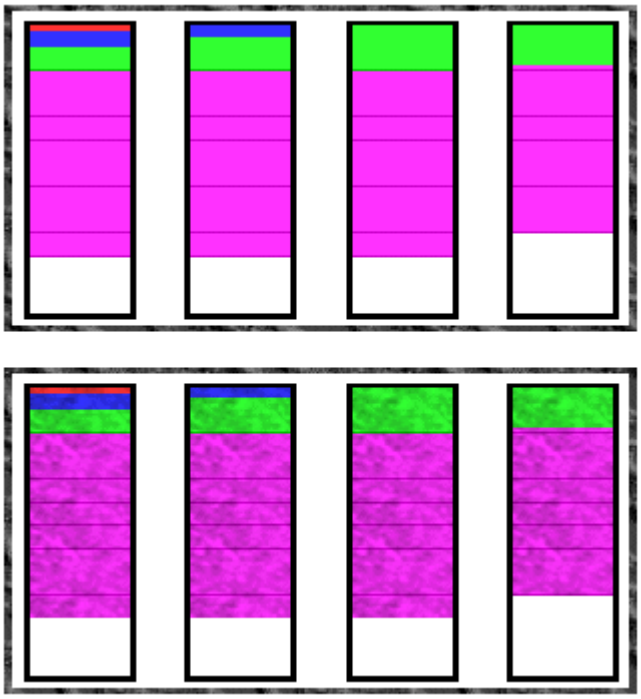
$$P_T = 1 - (1 - {p_n}^2)^5 \quad [2]$$

Esta última fórmula nos da la probabilidad total de falla de nuestro sistema. Se puede demostrar que [2] siempre es menor o igual que [1] mientras que P_n varíe entre 0 y 1. Por lo tanto, el segundo esquema, perteneciente a la configuración RAID 1+0 es más robusto que la primera configuración perteneciente a una configuración RAID 0+1.
Ahora que ya hemos analizado lo que implica tener configuraciones de RAID anidadas, procederemos a la descripción de cada una de ellas.

RAID 0+1 (01) y 1+0 (10)

- **Nombre con el que se lo conoce:** RAID 0+1, 01, 0/1, “mirrored stripes”, “mirror of stripes”; RAID 1+0, 10, 1/0, “striped mirrors”, “stripe of mirrors”.
- **Técnica utilizada:** Mirroring y striping sin paridad.

Descripción: Es el más popular de todos los niveles de RAID anidados, básicamente porque se obtiene lo mejor de las técnicas de striping y mirroring. Provee excelente niveles de performance y además es tolerante a fallos. Como ya vimos, el nivel 1+0 provee mejor tolerancia a fallos que el nivel 0+1, y además una mejor performance en la reconstrucción.



En este ejemplo se ve cómo los diferentes archivos (de igual tamaño que en los casos anteriores) se distribuyen entre 8 discos. Los dos rectángulos negros representan los sub arrays en RAID0, los cuales son duplicados utilizando la técnica de mirroring.

- **Requerimientos de la controladora:** Casi todas las controladoras del mercado soportan alguna de las dos configuraciones, pero muy pocas las dos. Las de baja gama generalmente soportan RAID01, mientras que las de alta gama generalmente soportan ambas.
- **Requerimientos de discos:** Número par de discos. Como mínimo 4.
- **Capacidad del array:** Tamaño del disco de menor capacidad por cantidad de discos sobre 2.
- **Eficiencia de almacenamiento:** Si todos los discos son del mismo tamaño, 50%.
- **Tolerancia a fallos:** Muy buena para RAID01, excelente para RAID10.
- **Disponibilidad de datos:** Muy buena para RAID01, excelente para RAID10.
- **Degradación y reconstrucción:** Pequeña para RAID10, puede llegar a ser sustancial en RAID01.
- **Performance de lecturas aleatorias:** Muy buena a excelente.
- **Performance de escrituras aleatorias:** Buena a muy buena.
- **Performance de lecturas secuenciales:** Muy buena a excelente.
- **Performance de escrituras secuenciales:** Buena a muy buena.
- **Costo:** Relativamente alto debido a la gran cantidad de discos y a la baja eficiencia de almacenamiento (50%).
- **Consideraciones especiales:** Debido a la baja eficiencia de almacenamiento condiciona la capacidad total del array.
- **Usos recomendados:** Aplicaciones que requieren tanto alta performance como tolerancia a fallos.

RAID 0+3 (03 ó 53) y 3+0 (30)

- **Nombre con el que se lo conoce:** El nombre con el que se conoce a este array es el más confuso de todos los niveles. Se lo debería conocer como RAID 03 o RAID 30, pero en lugar de 03 muchos usan el nombre 53. Y como si fuera poco, muchos implementan el 53 como 30 y no como 03. Entonces, como siempre y más que nunca, es bueno saber bien los conceptos para saber qué es lo que realmente el fabricante está implementando.

- **Técnica utilizada:** Striping a nivel de byte con paridad dedicada combinada con striping a nivel de bloque.

Descripción: Este nivel de RAID es combina los conceptos de striping a nivel de byte, paridad y striping a nivel de bloque. El nivel RAID 30 es más común que el RAID 03. El uso de la paridad, del striping de bloques pequeños (bytes) y el striping de bloques grandes hace que su análisis sea muy difícil. En general provee mejor performance que el RAID 3 debido al uso del striping del RAID 0, pero la misma se sitúa más cerca de la performance del RAID 3 que la del RAID 0, especialmente en las escrituras. El RAID 30 provee mejor tolerancia a fallos y mejor performance en la reconstrucción que el RAID 03, pero ambas dependen más del ancho del stripe del array en RAID 3 que el ancho del array en RAID 0.
Para este nivel de RAID daremos algunos ejemplos para algunas características analizadas, para que los conceptos queden más claros.

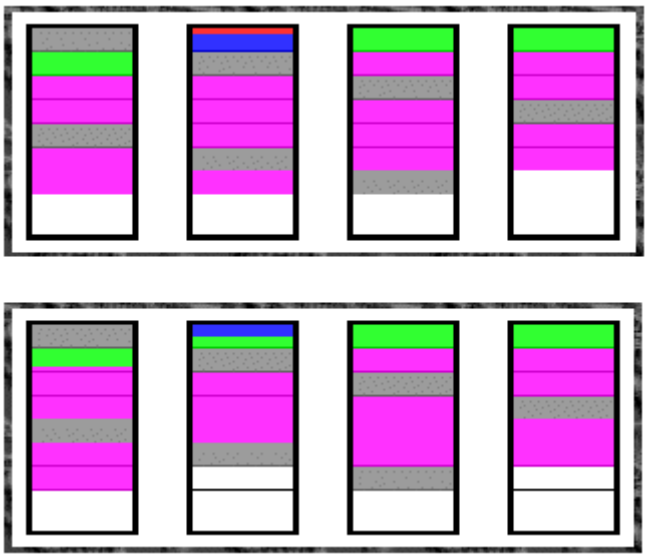
- **Requerimientos de la controladora:** Se requiere de una controladora de alta gama (alto costo)
- **Requerimientos de discos:** El número total de discos debe ser múltiplo de dos enteros, uno de ellos debe ser 2 ó más, y el otro 3 ó más. Por ejemplo, se podría tener un RAID 30 conformado por 10 discos pero no por 11). El mínimo es de 6 discos. El máximo depende de la controladora.
- **Capacidad del array:** Para RAID 03: (Capacidad del disco más chico)*(Número de discos en cada conjunto RAID 0)*(Cantidad de conjuntos RAID 0). Para RAID 30: (Capacidad del disco más chico)*(Número de discos en cada conjunto RAID 3 – 1)*(Cantidad de conjuntos RAID 3).
  Por ejemplo, tenemos un RAID 03 de 15 discos de 100 GB, configurados como 3 conjuntos de RAID 0 de 5 discos. La capacidad en este ejemplo es de 100 GB * 5 * (3-1) = 1 TB.
  Si tenemos un RAID 30 de 21 discos de 100 GB, configurados como 3 conjuntos de RAID 3 de 7 discos. La capacidad en este ejemplo es de 100 Gb * (7-1) * 3 = 1,8 TB.
- **Eficiencia de almacenamiento:** Para RAID 03: ((Cantidad de conjuntos RAID 0 – 1) / Cantidad de conjuntos RAID 0). Para RAID 30: ((Número de discos en cada conjunto RAID 3 – 1) / Número de discos en cada conjunto RAID 3).
  Ejemplo. Tomando los mismos ejemplos que en el caso anterior, la eficiencia en el arreglo de 15 discos en RAID 03 sería: (3-1) / 3 = 67%.
  En RAID 30, con el ejemplo anterior de 21 discos, la eficiencia sería: (7 – 1) / 7 = 86%.
- **Tolerancia a fallos:** Buena a muy buena, dependiendo de si la configuración es RAID 03 o RAID 30. RAID 30 provee mejor tolerancia a fallos.
- **Disponibilidad de datos:** Muy buena a excelente.
- **Degradación y reconstrucción:** Relativamente pequeña en RAID 30 (aunque más que en RAID 10), puede ser sustancialmente mayor en RAID 03.
- **Performance de lecturas aleatorias:** Muy buena, suponiendo un gran número de discos en el array RAID 0.
- **Performance de escrituras aleatorias:** Razonable.
- **Performance de lecturas secuenciales:** Muy buena a excelente.
- **Performance de escrituras secuenciales:** Buena.
- **Costo:** Relativamente alto debido a los requerimientos de la controladora y la cantidad de discos.
- **Consideraciones especiales:** Complejo y muy caro de implementar.
- **Usos recomendados:** Realmente se utiliza muy poco en la práctica. Se utiliza para aplicaciones que requieren de la velocidad de un RAID 0, con tolerancias a fallos y gran capacidad.

RAID 0+5 (05) y 5+0 (50)

- **Nombre con el que se lo conoce:** RAID 0+5 ó RAID 05 y RAID 5+0 ó RAID 50. Como siempre, es preferible ver bien lo que la controladora está implementado y no fiarse a ciegas por la nomenclatura.
- **Técnica utilizada:** Striping a nivel de bloque con paridad distribuida con striping a nivel de bloque sin paridad.

Descripción: Este nivel de RAID está conformado combinando la técnica de striping con paridad distribuida que ya vimos en RAID 5 con la técnica de striping sin paridad que vimos en el RAID 0. Ver el

RAID 50 es más común que el RAID 05. Con este nivel múltiple de RAID se obtiene una mejor performance que en RAID 5 debido a la incorporación del RAID 0 en él, sobre todo en las escrituras. Y también se obtiene una mejor tolerancia a fallos que en el nivel simple de RAID, sobre todo si se utiliza RAID 50 en lugar de RAID 05. Muchas de las características que vimos para los niveles RAID 03 y 30 son aplicables a este RAID, pero para ambientes transaccionales con archivos pequeños se prefieren estos niveles (05 y 50).



Merece la pena explicar este gráfico. Al igual que los anteriores, los tamaños de los archivos se mantienen. Es decir, el rojo es de 4 Kb, el azul de 20 Kb, el verde de 100 Kb y el magenta de 500 Kb. Cada bloque dentro de cada disco representa un tamaño de 16 Kb (el stripe size). Cada uno de los rectángulos grandes representa un array en RAID 5 de 4 discos cada uno. Los datos, se "stripean" entre estos dos arrays uniformemente, y por supuesto, dentro de los arrays en RAID 5 los datos son almacenados utilizando el striping con paridad distribuida. Por lo tanto, fíjense cómo el primer archivo de 4 Kb y los 12 Kb primeros Kb del segundo archivo se envían al primer array RAID 5, mientras que los 8 Kb restantes del segundo archivos más los primeros 8 Kb de los 100 Kb del tercero se envían al segundo array. Luego, los siguientes 16 Kb del archivo de 100 Kb se envían al primer array y así sucesivamente. Los datos en cada array se guardan de la misma manera que en RAID 5, pero como son dos los arrays, cada uno utiliza la mitad de los bloques que utilizaría si se utilizara un nivel de RAID 5 simple.

- **Requerimientos de la controladora:** Se requiere de una controladora de alta gama (alto costo)
- **Requerimientos de discos:** El número total de discos debe ser múltiplo de dos enteros, uno de ellos debe ser 2 ó más, y el otro 3 ó más.
- **Capacidad del array:** La misma que para los niveles RAID 03 y 30. Para RAID 05: (Capacidad del disco más chico)*(Número de discos en cada conjunto RAID 0)*(Cantidad de conjuntos RAID 0). Para RAID 50: (Capacidad del disco más chico)*(Número de discos en cada conjunto RAID 5 – 1)*(Cantidad de conjuntos RAID 5).
- **Eficiencia de almacenamiento:** La misma que para los niveles RAID 03 y 30. Para RAID 05: ((Cantidad de conjuntos RAID 0 – 1) / Cantidad de conjuntos RAID 0). Para RAID 50: ((Número de discos en cada conjunto RAID 5 – 1) / Número de discos en cada conjunto RAID 5).
- **Tolerancia a fallos:** Buena a muy buena, dependiendo de si la configuración es RAID 05 o RAID 50. RAID 50 provee mejor tolerancia a fallos.
- **Disponibilidad de datos:** Muy buena a excelente.
- **Degradación y reconstrucción:** Moderada para RAID 50, peor para RAID 05.
- **Performance de lecturas aleatorias:** Muy buena a excelente.
- **Performance de escrituras aleatorias:** Buena.
- **Performance de lecturas secuenciales:** Muy buena.
- **Performance de escrituras secuenciales:** Buena.

- **Costo:** Relativamente alto debido a los requerimientos de la controladora y la cantidad de discos.
- **Consideraciones especiales:** Complejo y muy caro de implementar.
- **Usos recomendados:** Aplicaciones que requieren una gran tolerancia a fallos, gran capacidad y buena performance de posicionamiento aleatorio.

RAID 1+5 (15) y RAID 5+1 (51)

- **Nombre con el que se lo conoce:** RAID 1+5 ó 15 y RAID 5+1 ó 51.
- **Técnica utilizada:** Mirroring y striping a nivel de bloque con paridad distribuida.

Descripción: Éste es el único nivel de RAID que utiliza ambas técnicas de redundancia, el mirroring y la paridad. Se utiliza principalmente para brindar una altísima tolerancia a fallos y disponibilidad de datos. El nivel RAID 15 se forma creando un conjunto que utiliza el striping con paridad distribuida usando varios pares de discos espejados como componentes. Es muy similar al nivel RAID 15, nada más que el striping se realiza con paridad. En cambio el nivel RAID 51 se forma utilizando dos arrays en RAID 5 y luego espejándolos, similar al RAID 01.
Lo que sobresale en este nivel múltiple de RAID es el nivel de tolerancia a fallos. Un nivel RAID 15 de 8 discos puede tolerar el fallo de 3 discos cualesquiera en todo el conjunto, y lo más sorprendente el nivel RAID 51, también con 8 discos puede tolerar también el fallo de 3 discos y hasta un fallo de 5, siempre y cuando al menos uno de los conjuntos en RAID 5 espejados no tenga más de un fallo. El costo de todo esto es la complejidad y la muy baja eficiencia de almacenamiento.

- **Requerimientos de la controladora:** Se requiere de una controladora de alta gama (alto costo). La mayoría de las controladoras del mercado no lo soportan, y por tanto, muchas veces se lo implementa como un híbrido, es decir, se toman dos array en RAID 5 creados por hardware y luego, a través del SO se implementa un mirroring entre ellos. Claro está, esto impacta muchísimo en la performance.
- **Requerimientos de discos:** Un mínimo par de 6 discos como mínimo y el máximo según la controladora.
- **Capacidad del array:** (Tamaño del menor disco) * ((Número de discos /2) – 1) / Número de discos.
- **Eficiencia de almacenamiento:** Asumiendo todos los discos como de igual capacidad. ((Número de discos / 2) – 1) / Número de discos.
- **Tolerancia a fallos:** Excelente, por lejos la mejor de todos los niveles.
- **Disponibilidad de datos:** Excelente.
- **Degradación y reconstrucción:** Puede ser sustancial.
- **Performance de lecturas aleatorias:** Muy buena.
- **Performance de escrituras aleatorias:** Buena.
- **Performance de lecturas secuenciales:** Muy buena.
- **Performance de escrituras secuenciales:** Buena.
- **Costo:** Muy alto.
- **Consideraciones especiales:** Complejo y muy caro de implementar.
- **Usos recomendados:** Aplicaciones críticas que requieren un altísimo nivel de tolerancia a fallos.

Just a Bunch Of Disks (JBOD)

JBOD en realidad no es un nivel de RAID, pero siempre que se estudian estos últimos el JBOD aparece entre mezclado, por lo tanto, ¿por qué no ponerlo también en este apunte?
Podríamos decir que JBOD es lo contrario a particionar. El particionamiento en los discos lo que hace es dividir a estos en volúmenes más pequeños. Contrariamente, JBOD lo que hace es tomar varios discos individuales y "juntarlos" como si fueran un solo volumen. La técnica utilizada tiene un nombre y se llama *"spanning"*. No provee fault tolerance ni ninguna mejora de performance. De hecho, la performance total es menor ya que no permite el acceso simultáneo a más de un disco. Para poder formar un JBOD se necesita una controladora o un driver por software al igual que con RAID0, nada más que éste último por lo menos provee una mejora en la performance. Hay dos ventajas posibles de JBOD sobre RAID 0:

- Permite el uso de varios discos de diferentes capacidades sin pérdida de capacidad. Un disco de 10 Gb y otro de 20 Gb con JBOD formarían un único volumen de 30 Gb, mientras que con RAID0 lo harían sólo formando un volumen de 20 Gb.
- Recuperación de desastre más fácil que con RAID0. En un RAID0 como los datos se "stripean" en todos los discos, el fallo de uno sólo hace que todos los datos en los demás archivos sean muy difícil de recuperar. En cambio con JBOD los datos de los demás discos se recuperarían más fácilmente, ya que no se utiliza el striping. Igualmente esto depende de cómo el hardware de la controladora o del software si JBOD se realiza por software manejan los diferentes discos.

Resumen de los diferentes niveles de RAID

A continuación mostramos el resumen del artículo original de los diferentes niveles de RAID.

RAID Level	Number of Disks	Capacity	Storage Efficiency	Fault Tolerance	Availability	Random Read Perf	Random Write Perf	Sequential Read Perf	Sequential Write Perf	Cost
0	2,3,4,...	S*N	100%	none	★	★★★★	★★★★	★★★★½	★★★★	$
1	2	S*N/2	50%	★★★★	★★★★	★★★	★★★	★★	★★★	$$
2	many	varies, large	~ 70-80%	★★	★★★★	★★	★	★★★★	★★½	$$$$$
3	3,4,5,...	S*(N-1)	(N-1)/N	★★★	★★★★	★★★	★	★★★★	★★½	$$
4	3,4,5,...	S*(N-1)	(N-1)/N	★★★	★★★★	★★★★	★½	★★★	★★	$$
5	3,4,5,...	S*(N-1)	(N-1)/N	★★★	★★★★	★★★★½	★★	★★★½	★★½	$$
6	4,5,6,...	S*(N-2)	(N-2)/N	★★★★½	★★★★★	★★★★½	★	★★★½	★★	$$$
7	varies	varies	varies	★★★	★★★★	★★★★½	★★★★	★★★★½	★★★★	$$$$$
01/10	4,6,8,...	S*N/2	50%	★★★★	★★★★½	★★★★½	★★★½	★★★★½	★★★½	$$$
03/30	6,8,9,10,...	S*N0*(N3-1)	(N3-1)/N3	★★★½	★★★★	★★★★	★★	★★★★½	★★★	$$$$
05/50	6,8,9,10,...	S*N0*(N5-1)	(N5-1)/N5	★★★½	★★★★	★★★★½	★★★	★★★★	★★★	$$$$
15/51	6,8,10,...	S*((N/2)-1)	((N/2)-1)/N	★★★★★	★★★★★	★★★★	★★★	★★★★	★★★	$$$$$

Funciones de RAID avanzadas

En este apartado explicaremos algunas funciones avanzadas referente a los sistemas RAID.

Caching

El caching o el uso de memorias cache es algo muy común en ambientes de computación. Las memorias cache hoy en día se encuentras en casi todos los componentes informáticos, y su función no es más que actuar como un buffer o un área de memoria que tiene como finalidad adaptar las velocidades de dos o más componentes de hardware.
Específicamente, mucha de las controladoras RAID tienen una memoria cache que evita que las operaciones de entrada y salidas tengan que ir directamente a los discos. Todos sabemos que la relación de velocidades entre el procesador o la memoria y los discos es de aproximadamente entre 100000 y 1000000 a 1. Cuando hablamos de accesos a memoria estamos hablando de demoras en torno a los nano segundos (10 a la -9 segundos), mientras que cuando dichos accesos los hacemos al disco dichas demoras rondan los mili segundos (10 a la -3 segundos). Por lo tanto, si en la controladora RAID se agrega una memoria intermedia de tipo cache, muchas de las operaciones de I/O se podrán resolver directamente en memoria (de la controladora) sin tener necesidad de acceso a los discos.
Así mismo es muy empleada la técnica de "write-back caching", donde la controladora le dice al SO que la operación de escritura finalizó tan pronto la misma es escrita en la memoria cache de la controladora, y luego éste realiza la escritura real en los discos, acelerando notablemente el tiempo de escritura.

Esta última técnica conlleva un gran riesgo que es la pérdida de datos o la discrepancia en los mismos, por ejemplo frente a la falla de energía. Vean que esta técnica "le dice" al SO que un dato fue escrito en los disco cuando realmente todavía no lo fue. Por lo tanto el SO continúa su ejecución "pensando" que el dato está escrito en memoria secundaria. Si en ese instante la energía de corta del sistema, ese dato nunca va a poder recuperarse. Es por esto que la mayoría de las controladoras del mercado que permiten el uso de esta técnica poseen una batería que permite mantener los datos en la memoria de la controladora hasta que la energía se restablezca y los datos puedan ser volcados a los discos.
Esta técnica presenta las mayores mejoras en sistemas RAID donde la escritura es penalizada, por ejemplo en RAID5 por la obligación de tener que escribir en todos los discos cada vez que se realiza una escritura.

Swapping

El swapping no es más que el reemplazo de un disco fallado por uno sano. Dependiendo de cuándo y bajo qué condiciones se puede realizar ese reemplazo, se definen tres tipos de swapping:

- **Hot Swap:** El verdadero hot swap es aquel que permite que el disco sea reemplazado mientras que todo el sistema permanece activo y en línea, es decir, sin interrupción alguno y dando el servicio como si no hubiera ningún fallo (excepto claro está, que el sistema ya está trabajando en modo degradado, pero funcionando y en servicio)
- **Warm Swap:** El warm swap permite realizar el reemplaza mientras que el sistema está encendido, pero con la necesidad de parar toda actividad en los buses de los dispositivos a los que está conectado el dispositivo. Claramente esto hace que el sistema no pueda brindar servicio, pero permanece encendido. Por lo tanto, es peor que el hot swap pero mejor que el cold swap que veremos a continuación.
- **Cold Swap:** En este caso, para efectuar el reemplazo es necesario apagar todo el sistema.

Hot Spares

Como ya hablamos, la funcionalidad de hot spare permite dejar uno o más discos como *spares* o repuesto. Estos discos están conectados al sistema pero en modo stand by, es decir, sin usar. Cuando cualquier disco de cualquier array falla (excepto en el RAID 0 que no brinda redundancia), el sistema desconecta de ese array el disco fallado y lo reemplaza automáticamente por el que estaba en modo hot spare. Esto hace que automáticamente el sistema para ese array entre en modo de reconstrucción.

Array Expansion

Esta funcionalidad lo que permite es que se puedan expandir con el sistema en servicio el tamaño de los arrays. La manera de hacer esto es agregar nuevos discos al sistema físicamente y luego expandir el tamaño del array en cuestión lógicamente. La mayoría de las controladoras de hoy en día permite esta operatoria.