https://repository.uaeh.edu.mx/revistas/index.php/huejutla/issue/archive

Ciencias Huasteca Boletín Científico de la Escuela Superior de Huejutla

Ciencia Huasteca
Boletín Científico de la Escuela Superior de Huejutla

Publicación semestral No. 14 (2019) 33-43

Sistema de archivos en Linux

File system in Linux

Angélica Carrión [a], Germán Álvarez [b], Cristhian Briones [b], Ingrid Ortiz [b], Paola Morales [b], José Cordero [b], Leslie Sinchiguano [b]

Abstract:

The present work has as objective to show a revision of the System of files of the computer system. For this reason, we refer to aspects of storage management, to RAID configurations that are storage systems since it entails redundant information for the local storage error. Likewise, the operation of the PVFS storage system is described, which allows a distribution to be made between different nodes that intervene in the storage of a data processing center. Finally, the formats used in the Linux and Windows operating systems are mentioned.

Keywords:

File system, RAID, PVFS, Linux.

Resumen:

El presente trabajo tiene como objetivo mostrar una revisión del Sistema de archivos del sistema informático. Por tal motivo, se describen aspectos de gestión de almacenamiento, las configuraciones RAID que son sistemas de almacenamiento existentes que conlleva a redundar información para tolerar fallas en el almacenamiento local. De igual manera, se describe el funcionamiento del sistema de almacenamiento PVFS que permite hacer una distribución entre diferentes nodos que intervienen en el almacenamiento de un centro de procesamiento de datos. Por último, se mencionan los formatos utilizados en los sistemas operativos Linux y Windows.

Palabras Clave:

Sistema de archivos, RAID, PVFS, Linux.

Introducción

Un sistema de archivo permite almacenar de una manera ordenada información que poseemos, como imágenes, documentos, etc., pero para realizar esto debe seguir un conjunto de normas o restricciones, todo Sistema Operativo (SO) posee un sistema de archivos, por ejemplo, en Windows el sistema de archivos predeterminado es NTFS (*New Technology File System*) y en Linux existen diferentes opciones. Un sistema operativo reconoce 3 diferentes tipos de archivos, como:

- Normales: estos archivos pueden contener cualquier tipo de información.
- *Directorios*: este tipo de archivos almacena información de otros archivos, utilizado para agrupar los archivos libremente en un sistema de archivos.
- *Especiales*: son archivos que permiten comunicarse con dispositivos de entrada/salida.

Poseen características en común como el nombre, la extensión del archivo, tipo de archivo, ubicación, permisos y atributos.
Todos los dispositivos de almacenamiento usan un sistema de archivos. Recordemos que un dispositivo de almacenamiento es tanto un disco duro como un CD o una memoria USB. Los sistemas de archivos de los dispositivos removibles pueden ser leídos por cualquier sistema operativo, como Windows o

[a] Autor de Correspondencia, Universidad Técnica Estatal de Quevedo, Facultad de Ciencias de la Ingeniería, Email: angelicangon.carrion@uteq.edu.ec

[b] Universidad Técnica Estatal de Quevedo, Facultad de Ciencias de la Ingeniería, Email: alvarezcarpiogerman@gmail.com, bcristhian5@gmail.com, ingrid.toli7@gmail.com, paolasofia17.pm@gmail.com, im_josecordero@outlook.com, leslie-alexander97@hotmail.es

GNU/Linux. Sin embargo, los sistemas de archivos nativos no siempre se pueden leer entre sistemas operativos. Por ejemplo, muchas distribuciones GNU/Linux pueden leer el sistema de archivos de Windows, pero Windows no puede leer ninguno de los sistemas que utiliza GNU/Linux.

¿Qué es un sistema de archivos?

Un sistema de archivos es un conjunto de normas y procedimientos para almacenar la información.

Un archivo es un conjunto independiente de datos, como una foto o un texto. Hay diferentes tipos de archivo como veremos más adelante. Toda la información que hay en una computadora está agrupada en forma de archivos [1].

Cada sistema operativo suele usar un sistema de archivos diferente. Pero todos comparten otro concepto: la carpeta. Una carpeta es una manera de agrupar libremente archivos. Las carpetas también se conocen como directorios.

Al sistema de archivos que utiliza un sistema operativo se le llama sistema de archivos nativo. Esa es la manera en que un SO prefiere guardar la información.
El proceso de crear un sistema de archivos en un dispositivo de almacenamiento se llama formatear o dar formato: preparar el dispositivo para guardar la información como lo hace el sistema elegido. Si formateamos un dispositivo, se borrará toda la información que está en él [1].

Todos los dispositivos de almacenamiento usan un sistema de archivos. Recordemos que un dispositivo de almacenamiento es tanto un disco duro como un CD o una memoria USB. Los sistemas de archivos de los dispositivos removibles pueden ser leídos por cualquier sistema operativo, como Windows o GNU/Linux.

Sin embargo, los sistemas de archivos nativos no siempre se pueden leer entre sistemas operativos. Por ejemplo, muchas distribuciones GNU/Linux pueden leer el sistema de archivos de Windows, pero Windows no puede leer ninguno de los sistemas que utiliza GNU/Linux.

Gestión de almacenamiento en disco

El segundo componente importante en cualquier computadora, luego del procesador, es la memoria. En teoría, una memoria debe ser en extremo rápida (más rápida que la velocidad de ejecución de una instrucción, de manera que la memoria no detenga a la CPU), de gran tamaño y muy económica.

El almacenamiento en disco es dos órdenes de magnitud más económica que la memoria RAM por cada bit, y a menudo es dos órdenes de magnitud más grande en tamaño también. El único problema es que el tiempo para acceder en forma aleatoria a los datos en ella es de cerca de tres órdenes de magnitud más lento.

Esta baja velocidad se debe al hecho de que un disco es un dispositivo mecánico.

Hacer que el sistema de archivos funcione es una cosa; hacerlo que funcione de manera eficiente y robusta en la vida real es algo muy distinto. En las siguientes secciones analizaremos algunas de las cuestiones involucradas en la administración de discos [2].

Administración del espacio en disco

Por lo general los archivos se almacenan en disco, así que la administración del espacio en disco es una cuestión importante para los diseñadores de sistemas de archivos. Hay dos estrategias generales posibles para almacenar un archivo de *n* bytes: se asignan *n* bytes consecutivos de espacio en disco o el archivo se divide en varios bloques (no necesariamente) contiguos. La misma concesión está presente en los sistemas de administración de memoria, entre la segmentación pura y la paginación.

Como hemos visto, almacenar un archivo como una secuencia contigua de bytes tiene el problema obvio de que, si un archivo crece, probablemente tendrá que moverse en el disco. El mismo problema se aplica a los segmentos en memoria, excepto que la operación de mover un segmento en memoria es rápida, en comparación con la operación de mover un archivo de una posición en el disco a otra. Por esta razón, casi todos los sistemas de archivos dividen los archivos en bloques de tamaño fijo que no necesitan ser adyacentes [3].

Tamaño de bloque

Una vez que se ha decidido almacenar archivos en bloques de tamaño fijo, surge la pregunta acerca de qué tan grande debe ser el bloque. Dada la forma en que están organizados los discos, el sector, la pista y el cilindro son candidatos obvios para la unidad de asignación (aunque todos ellos son dependientes del dispositivo, lo cual es una desventaja). En un sistema de paginación, el tamaño de la página también es uno de los principales contendientes.

Registro de bloques libres

Una vez que se ha elegido un tamaño de bloque, la siguiente cuestión es cómo llevar registro de los bloques libres. Hay dos métodos utilizados ampliamente: *a)* una lista enlazada y *b)* mapa de bits, tal como se muestra en la Figura 1. En este caso se explica el primero, el cual, consiste en utilizar una lista

enlazada de bloques de disco, donde cada bloque contiene tantos números de bloques de disco libres como pueda. Con un bloque de 1 KB y un número de bloque de disco de 32 bits, cada bloque en la lista de bloques libres contiene los números de 255 bloques libres (se requiere una ranura para el apuntador al siguiente bloque).

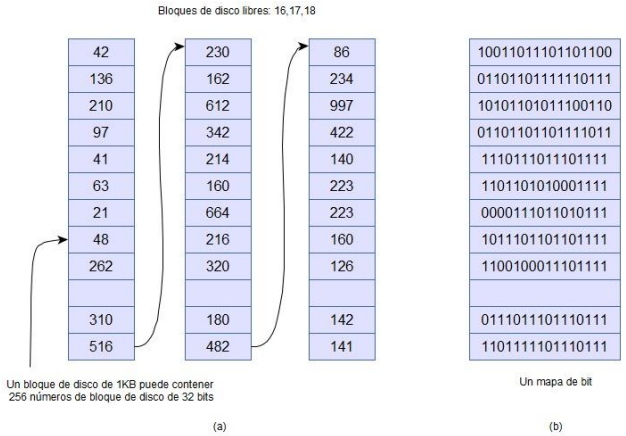

Figura 1. Registro de bloques libres

Considere un disco de 500 GB, que tiene aproximadamente 488 millones de bloques de disco. Para almacenar todas estas direcciones a 255 por bloque, se requiere una cantidad aproximada de 1.9 millones de bloques. En general se utilizan bloques libres para mantener la lista de bloques libres, por lo que en esencia el almacenamiento es gratuito.

Cuotas de disco

Para evitar que los usuarios ocupen demasiado espacio en disco, los sistemas operativos multiusuario proporcionan un mecanismo para imponer las cuotas de disco. La idea es que el administrador del sistema asigne a cada usuario una cantidad máxima de archivos y bloques y que el sistema operativo se asegure de que los usuarios no excedan sus cuotas. A continuación, describiremos un mecanismo común.

Cuando un usuario abre un archivo, los atributos y las direcciones de disco se localizan y se colocan en una tabla de archivos abiertos en la memoria principal. Entre los atributos hay una entrada que indica quién es el propietario. Cualquier aumento en el tamaño del archivo se tomará de la cuota del propietario.

Una segunda tabla contiene el registro de cuotas para cada usuario con un archivo actualmente abierto, aun si el archivo fue abierto por alguien más. Esta tabla se muestra en la Figura 2. Es un extracto de un archivo de cuotas en el disco para los usuarios cuyos archivos están actualmente abiertos. Cuando todos los archivos se cierran, el registro se escribe de vuelta en el archivo de cuotas.

Cuando se crea una nueva entrada en la tabla de archivos abiertos, se introduce en ella un apuntador al registro de cuotas del propietario, para facilitar la búsqueda de los diversos límites. Cada vez que se agrega un bloque a un archivo se incrementa el número total de bloques que se cargan al propietario y se realiza una verificación con los límites duro y suave. Se puede exceder el límite suave, pero el límite duro no. Un intento de agregar datos a un archivo cuando se ha llegado al límite de bloques duro producirá un error. También existen verificaciones similares para el número de archivos.

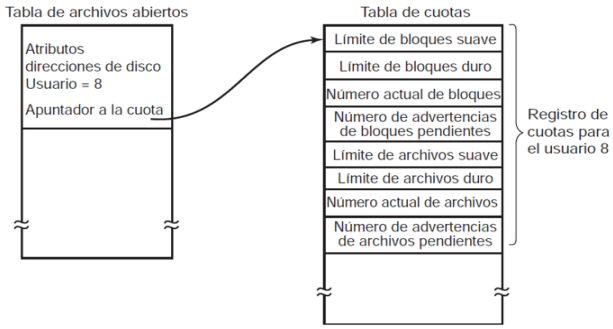

Figura 2. Cuotas de disco

Cuando un usuario trata de iniciar sesión, el sistema examina el archivo de cuotas para ver si el usuario ha excedido el límite suave para el número de archivos o el número de bloques de disco.
Si se ha violado uno de los límites, se muestra una advertencia y la cuenta de advertencias restantes se reduce en uno. Si la cuenta llega a cero en algún momento, el usuario ha ignorado demasiadas veces la advertencia y no se le permite iniciar sesión. Para obtener permiso de iniciar sesión otra vez tendrá que hablar con el administrador del sistema.

Este método tiene la propiedad de que los usuarios pueden sobrepasar sus límites suaves durante una sesión, siempre y cuando eliminen el exceso cuando cierren su sesión. Los límites duros nunca se pueden exceder [3].

RAID (*Redundant Array of Independent Disks*)

El sistema RAID (ver Figura 3), que significa arreglo redundante de discos independientes o de bajo costo. Además de proveer tolerancia a fallos, se utiliza para brindar un mejor rendimiento, que se logra a partir de la escritura en varios discos al mismo tiempo. El arreglo de discos se puede realizar tanto por hardware con controladoras RAID como por software, y existen varios tipos de RAID según cómo se realiza. Estos tipos se denominan con números de niveles y se utilizan para obtener distintos beneficios, como puede ser rendimiento en algunos Casos, tolerancia a fallos en Otros, 0 ambos.

Para configurar un sistema RAID se emplean varios discos, generalmente de la misma capacidad. El sistema operativo muestra todos los discos que forman parte del arreglo como un solo disco, y no se alcanza la misma capacidad total que tendríamos si usáramos los discos por separado. Esto sucede porque, para es guardar los datos, éstos se duplican o se escriben por partes en diferentes discos.

Hardware de servidor perdido a causa de la falla de un disco se puede recuperar a partir de la reconstrucción de los datos restantes y de una información adicional redundante que se guarda por cada bloque de información, llamada paridad. Esta paridad luego es utilizada para recuperar los datos que se pierden.
Existen varios tipos de RAID, y cada uno de ellos se adecua a diferentes necesidades y recursos.

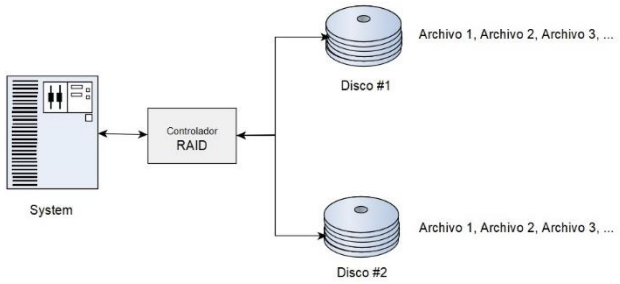

Figura 3. Sistema RAID

RAID 0

Este arreglo no garantiza la protección de los datos porque no tiene tolerancia a fallos (no guarda datos de forma redundante). Su utilidad es permitir una velocidad más alta en operaciones de escritura y de lectura, porque utiliza más de un disco a la vez. Es muy útil para sistemas que realizan muchas operaciones con archivos grandes, pero no se utiliza en servidores. Requiere como mínimo dos discos para implementarse (Figura 4).

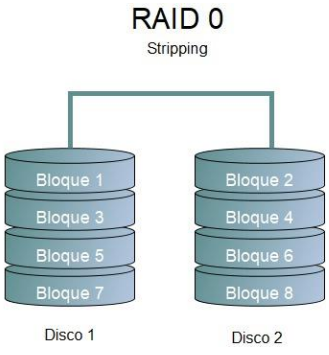

Figura 4. Configuración RAID 0

RAID 1

Este RAID utiliza dos discos (ver Figura 5) de igual tamaño y duplica toda la información en ambos, es decir, un disco es el espejo del otro. Aunque esto provee un mecanismo muy seguro para resguardar los datos, su desventaja es que utiliza mucho espacio para almacenar todo duplicado. Necesita como mínimo dos discos para implementarse, y pueden utilizarse varios pares de discos espejados. El espacio disponible de este arreglo será la mitad de la capacidad total de los discos que utilicemos para armarlo.

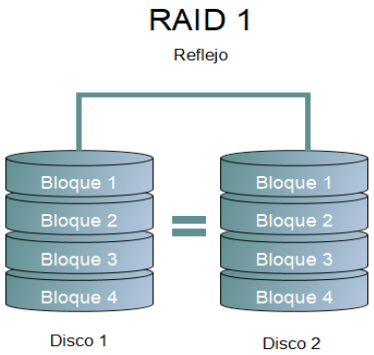

Figura 5. Configuración RAID 1

RAID 2

Este nivel de RAID separa la información en bits que escribe en cada disco y además utiliza discos adicionales para almacenar un código de corrección de errores (Figura 6). Puede reparar información da nada mediante el código ECC (*Error Correction Code*), que almacena con un método llamado Hamming.

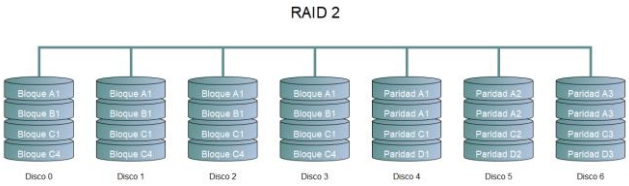

Figura 6. Configuración RAID 2

RAID 3

Este RAID usa un disco de comprobación (denominado de paridad) para cada arreglo de discos. Divide en partes los bloques de información, las escribe en paralelo, y graba en ese disco de paridad una información que puede ser recuperada en caso de fallas. Utiliza como mínimo tres unidades de disco (Figura 7).

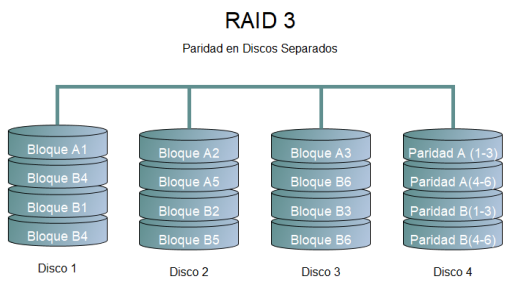

Figura 7. Configuración RAID 3

RAID 4

Este RAID es similar al RAID 3, pero en lugar de dividir los bloques en partes para escribirlos, los envía completos a cada disco, y también guarda información de paridad en el disco adicional (Figura 8). Necesita un mínimo de tres discos para su implementación.

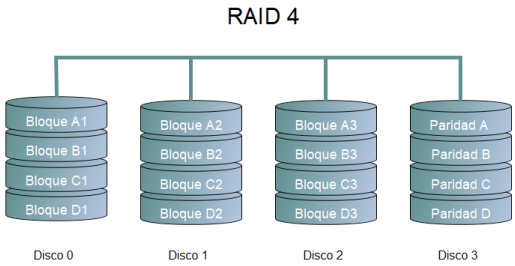

Figura 8. Configuración RAID 4

RAID 5

Este RAID almacena la información en bloques y guarda información de paridad (Figura 9), pero a diferencia de los anteriores, no utiliza un disco especial para hacerlo, sino que va guardando ese dato en todos los discos. Esto evita que, si se arruina el disco de paridad, resulte imposible recuperar la información.

Para implementarlo, se requieren como mínimo 3 discos. En comparación con el nivel 1, en caso de fallas, es más difícil recuperar la información, pero no se desaprovecha tanto espacio. La capacidad total que se obtiene con este arreglo es la suma de la capacidad de todos los volúmenes menos el equivalente a uno.

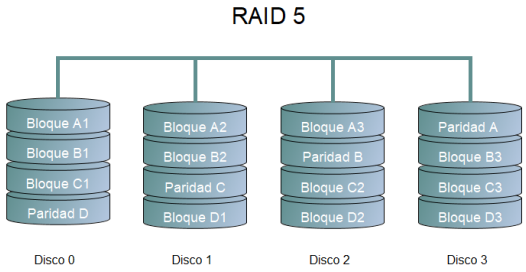

Figura 9. Configuración RAID 5

RAID 6

Este nivel de RAID es, básicamente, igual al nivel 5, pero con información de paridad adicional que provee tolerancia a fallos frente a la falla de dos discos (Figura 10). Requiere un mínimo de cuatro discos, y no es uno de los sistemas RAID originales.

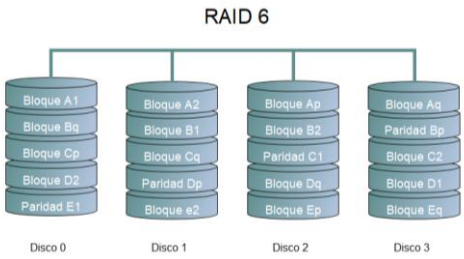

Figura 10. Configuración RAID 6

RAID 10

Este nivel es uno de los tipos anidados, lo que significa que combina dos niveles de RAID (Figura 11). Por un lado, implementa RAID l, del que hereda la velocidad y la tolerancia a fallos (por la duplicación). Del nivel O, toma la división de la información en varias unidades de disco. Este nivel híbrido es ideal para sistemas con altos requerimientos y tolerancia a fallos. Sólo debemos tener en cuenta que necesita un mínimo de cuatro discos para funcionar.

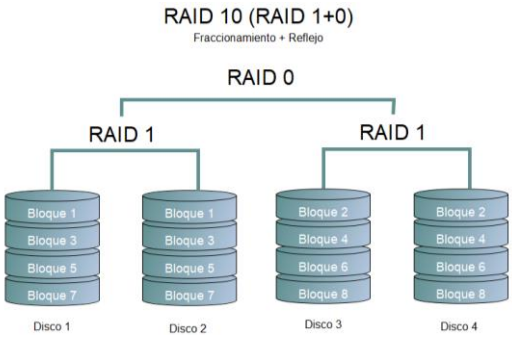

Figura 11. Configuración RAID 10

A manera de resumen, en la Tabla 1 se presente una descripción detallada de las configuraciones RAID disponibles. En dicha tabla se considera el aumento de velocidad, número mínimo de discos, la capacidad, si tiene o no tolerancia a fallos y la concurrencia.

Nivel	Aumento de velocidad	N# discos (mínimo)	Capacidad	Fallos tolerancia	concurrencia
RAID 0	Alto, crece linealmente con el número de discos.	2	N discos	ninguno	no guarda datos de forma redundante
RAID 1	Alto en lectura si la controladora permite paralelizar, igual o peor en escritura debido a duplicación de los datos.	2	Muy poco eficiente, Se recomienda el uso de dos discos.	N-1	La duplicación de datos mejora la velocidad de Lectura en accesos concurrentes.
RAID 0+1	Similar a RAID 0	4	Igual a RAID 1, es decir la capacidad total equivale a N/2 discos	Como mínimo 1, posiblemente más dependiendo de su distribución	Similar a RAID 1
RAID 2	Alto en lectura similar a RAID 0. Escritura lenta al tener que calcular los códigos de Hamming. Ralentización adicional por tener que tratar a nivel de bit.	Enorme, tantos como bits tenga el tamaño de palabra.	Gran sobrecarga, ya que son necesarios log2(M)+1 discos adicionales.	1	Sin aumento especial de rendimiento.
RAID 3	Similar a RAID 0 en la lectura. Disminución en escritura al tener que calcular la paridad, el disco de paridad es un cuello de botella.	3	Buena eficiencia en almacenamiento, la capacidad del sistema es equivalente a N-1 discos.	1	Muy malo, ya que al funcionar a nivel de byte cualquier bloque se encuentra distribuido a lo largo de todo el RAID.
RAID 4	Superior a RAID 3 al trabajar con bloques. El disco de paridad sigue siendo un cuello de botella.	3	Idéntica a RAID 3	1	Mejor que RAID 3 al no requerir accesos a todos los discos para cada petición (basta acceder al disco que tiene el bloque).
RAID 5	Superior a RAID 4 al eliminar el cuello de botella del disco de paridad.	3	Idéntica a RAID 3 y 4.	1	Mejor que RAID 3 y 4 al no haber cuellos de botella. Además, puede incluso llegar a ser más rápidos.
RAID 6	Idéntica a RAID 5 en lectura, escrituras algo más lento al tener que escribir 2 veces la paridad.	4	Menos eficiente que RAID 5 debido a la doble paridad. La capacidad total del sistema es equivalente a N-2 discos	2	El comportamiento es mejor que RAID 5 al estar los datos más dispersos por la obligación de usar un disco adicional

Tabla 1. Características de las configuraciones RAID

Sistemas de archivos paralelos

Un sistema de archivos paralelo es aquél que elimina el problema del cuello de botella de E/S, agregando de forma lógica múltiples dispositivos de almacenamiento independientes y nodos de E/S mediante un sistema de almacenamiento de alto rendimiento [4].

- Direccionamiento de disco independiente, mediante el cual el sistema de ficheros puede acceder a datos de diferentes ficheros de forma concurrente.
- Reparto de los datos en los nodos, mediante el cual un solo fichero se puede acceder en paralelo.

GPFS (*General Parallel File System*)

Es un sistema de ficheros desarrollado por IBM. GPFS es particularmente apropiado en entornos donde los sistemas distribuidos no ofrecen suficiente rendimiento de ancho de banda. Siendo un entorno que permite a los usuarios compartir el acceso a los datos a través de clúster, posibilitando la interacción a través de las interfaces estándar de UNIX [5]. La fortaleza de GPFS se basa en:

- Mejora el rendimiento del sistema: permite que múltiples nodos accedan simultáneamente a los datos utilizando llamadas estándar del sistema, balanceando la carga y por lo tanto incrementando el ancho de banda disponible por cada nodo.
- Asegura la consistencia de los datos: utiliza un sofisticado sistema de administración que provee la consistencia de los datos mientras permite múltiple e independientes rutas para archivos con el mismo nombre.
- Alta recuperabilidad y disponibilidad de los datos: crea registros logs separados para cada uno de los nodos que intervienen en el sistema. Permite administrar el número de réplicas que se desea manejar.
- Alta flexibilidad del sistema: los recursos no se encuentran congelados, se puede añadir o quitar discos al sistema mientras este se encuentra montado. Cuando la demanda es muy baja se puede reconfigurar la cargar del sistema a través de todos los discos configurados. También se puede agregar nuevos nodos sin que el sistema sea detenido y puesta en marcha nuevamente.
- Administración simplificada: los comandos de GPFS guardan la configuración en más de un archivo, conocido colectivamente como clúster de datos. Los comandos de GPFS están diseñados para sincronizar los datos en cada uno de los nodos del sistema. De tal modo se asegura una exacta configuración de los datos.

PVFS (*Parallel Virtual File System*)

PVFS admite E/S paralelas en Linux ampliamente utilizado en la informática de clúster, se desarrolló en la Universidad de Clemson. PVFS se compone de nodos de procesamiento, administrador de metadatos único y servidores de E/S.

Los nodos de procesamiento son clientes que usan servicios PVFS. El administrador de metadatos administra los metadatos de los archivos PVFS. Los servidores de E/S almacenan los datos de los archivos PVFS.

En PVFS, los datos de un archivo se eliminan en los servidores de E/S. Hay dos esquemas mediante los cuales los usuarios pueden acceder a archivos en PVFS. En primer lugar, los usuarios pueden acceder a los archivos mediante la recompilación de sus códigos de aplicación de aplicaciones con el archivo de nivel de usuario de PVFS, el esquema de la biblioteca PVFS. Otro esquema es, donde los usuarios pueden acceder a los archivos a través de la llamada al sistema de E/S de UNIX utilizando el módulo Kernel.

Hubo una investigación para el efecto de caché del sistema de archivos de PVFS donde Vilayannur et al., diseñó e implementó una memoria caché del sistema de archivos de un cliente en el esquema de la biblioteca PVFS. Mostraron que un caché de sistema de archivos en un cliente es eficiente si muchas aplicaciones en el cliente comparten archivos entre ellos. Pero su investigación se limitó a un caché de sistema de archivos en un solo nodo. Debido a que muchos usuarios comparten archivos en entornos de clúster, la memoria caché cooperativa es más apropiada que la caché de un sistema de archivos en un cliente.

Coopc-PVFS

En el esquema del módulo de Kernel PVFS, una aplicación lee un archivo de servidores de E/S a través del módulo de Kernel PVFS sin una función de caché del sistema de archivos. Se agrega el almacenamiento en caché colaborativa al esquema de módulo Kernel PVFS en nodos de procesamiento. En la Figura 12 se muestra el flujo de trabajo de Coopc-PVFS agregado a la arquitectura de PVFS.

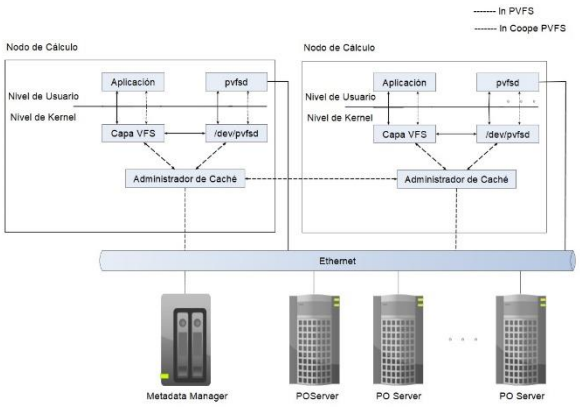

Figura 12. Arquitectura de la implementación de PVFS

En la caché colaborativa agregada al esquema del módulo de Kernel de PVFS, cuando una aplicación lee un archivo, el administrador de caché del cliente busca si el bloque solicitado está en su propia caché. Si el bloque está en el administrador de caché, el administrador de caché copia el bloque en la aplicación. Si el bloque no se encuentra en su propio administrador de caché, el administrador de caché busca si hay algún cliente que almacene en caché el bloque. Si el bloque es redondo en los administradores de caché de los otros clientes, el administrador de caché obtiene el bloque de uno de ellos, y luego almacena en caché el bloque y lo copia a la aplicación. Si el bloque no se encuentra en el administrador de caché de otros clientes, el administrador de caché obtiene el bloque de los servidores de E/S, y luego almacena en caché el bloque y lo copia en la aplicación.

Debido a la gran sobrecarga para mantener la información precisa sobre los bloques en caché, se ha diseñado Coopc-PVFS como un caché cooperativo basado en sugerencias. Para mantener la lista de clientes abiertos indirectos, se agrega una nueva función al administrador de metadatos para mantener la lista de clientes que contiene información de clientes que abrieron el archivo anteriormente. Cada vez que un cliente abre un archivo, el cliente obtiene los metadatos y la lista de clientes abiertos del archivo desde el administrador de metadatos.
Para buscar con precisión un bloque si otros clientes lo tienen, el cliente debe conocer la información sobre los bloques almacenados en caché en otros clientes.

En PVFS, todos los accesos a los archivos pasan por los servidores de E/S. Para conservar la coherencia en PVFS, el administrador de caché debe invalidar los bloques almacenados en caché en otros clientes antes de escribir el bloque en el servidor de E/S en Coopc-PVFS. Para hacer eso, cada vez que una aplicación escribe un bloque, el administrador de caché envía la solicitud de propagación de invalidación de bloque al administrador de metadatos antes de enviar el bloque escrito al servidor de E/S. Cuando el administrador de metadatos obtiene la solicitud de propagación de invalidación de bloque, envía los

mensajes de invalidación de bloque a los clientes en la lista de clientes abiertos del archivo, luego todos los clientes que reciben el mensaje de invalidación de bloque invalidan el bloque. Por lo tanto, todos los bloques en caché en otros clientes se invalidan antes de enviar el bloque escrito al servidor de E/S y así se preserva la coherencia en CoopcPVFS de la misma manera que en PVFS.

La implementación de Coopc-PVFS se muestra en la Figura 13 en la que se presentan las estructuras de datos utilizadas por el administrador de caché en Coopc-PVFS.

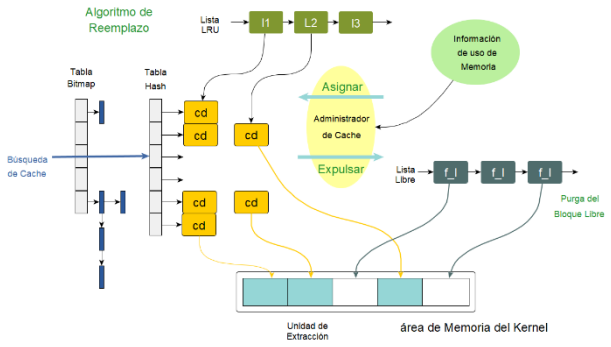

Figura 13. Algoritmo de reemplazo de caché en Coopc-PVFS

Sistemas de archivos de Linux

En Linux existen sistemas de archivos que se pueden utilizar, tales como: ext2, ext3, ext4, XFS, ReiserFS, etc. En esta sección se hace una descripción particular del sistema de archivos local ext3.

Tipos de archivos

- Normales: los archivos pueden contener cualquier tipo de información en algunos casos pueden ser programas que una vez ejecutados pasan a ser procesos. Dentro de los archivos normales podemos encontrar archivos ejecutables de audio, de imágenes, video, comprimidos, etc.
- Directorios: son archivos que contienen información sobre la organización y estructura de otros archivos.
- Especiales: estos archivos permiten comunicarse con dispositivos de E/S (archivos especiales de caracteres) o con discos (archivos especiales de bloque).

Propiedades

Algunas propiedades que tienen los archivos.

- Nombre: sirve para identificar un archivo, la mayoría de los sistemas operativos permiten utilizar nombres de hasta 255 caracteres, algunos sistemas operativos hacen distinciones entre mayúsculas y minúsculas.
- Extensión: la extensión sirve para saber el tipo de programa que lo ejecuta o interpreta, algunos sistemas operativos como Windows utilizan extensiones en otros como Unix o Linux, aunque se pueden utilizar.
- Tipo de archivo: pueden dividirse en normales, directorios o especiales.
- Ubicación: lugar del sistema de archivos donde se encuentra un archivo.
- Fecha de creación, modificación, y último acceso: para cada archivo indica la fecha en que ha sido creado, modificado, y la última vez que se accedió.
- Atributos: son propiedades que permiten asignar características especiales a los archivos, algunos ejemplos son atributos de solo lectura, de archivos de sistema, ocultar, cifrar comprimir, etc.
- Permisos: se utilizan para permitir o restringir el acceso a los archivos a determinados usuarios o grupos. Los permisos pueden ser para leer, modificar, eliminar, renombrar, etc., [6].

En cuanto a los sistemas de archivos soportados por la mayoría de las distribuciones Linux, la versión ext3 es una mejora de la ext2, con algunas características nuevas, como el *journaling*, y el ext4 es una mejora de la ext3, con más mejoras como soportar volúmenes y ficheros más grandes, mejora de fiabilidad, rapidez, aunque los tres sistemas de archivos son compatibles entre sí.

El *journaling* es una técnica en la que se registra a diario los cambios en el sistema de archivos para poder recuperar los datos en caso de fallo [7].

El sistema de archivos ext3 ofrece la posibilidad de trabajar con *journaling*, sistema mediante el cual se salvan periódicamente los archivos abiertos a fin de evitar la pérdida de información o la corrupción de datos si se produce una desconexión no planificada. Este sistema de archivos aporta más seguridad, a pesar de que por el contrario hace perder recursos de máquina, asignados precisamente a la tarea de *journaling* [8].

Características

Básicamente, el sistema de archivos ext3 es una versión mejorada de ext2. Las mejoras introducidas proporcionan las siguientes ventajas:

- Disponibilidad tras un corte eléctrico o una caída inesperada del sistema (también se denomina cierre no limpio del sistema), se debe comprobar con el programa e2fsck cada sistema de archivos ext2 montado en la máquina para ver si es consistente. El proceso de comprobación lleva mucho tiempo y puede prolongar el tiempo de arranque del sistema de un modo significativo, especialmente si hay grandes

volúmenes que contienen un elevado número de archivos. Durante este proceso, no se puede acceder a los datos de los volúmenes.

Con la característica *journaling* del sistema de archivos ext3 ya no es necesario realizar este tipo de comprobación en el sistema de archivos después de un cierre no limpio del sistema. En el sistema ext3, únicamente se realiza una comprobación de consistencia en los casos puntuales en los que se producen determinados errores de hardware, como, por ejemplo, fallos en el disco duro. El tiempo empleado para recuperar un sistema de archivos ext3 tras un cierre no limpio del sistema no depende del tamaño del sistema de archivos ni del número de archivos, sino del tamaño del *journal* (diario), utilizado para mantener la consistencia en el sistema. Por defecto, la recuperación del tamaño del "*journal*" tarda alrededor de un segundo, según la velocidad del hardware.

La integridad de los datos en el sistema de archivos ext3 proporciona una integridad superior de los datos si se produce un cierre no limpio del sistema. El sistema de archivos ext3 le permite seleccionar el tipo y el nivel de protección de los datos. Por defecto, los volúmenes ext3 son configurados para mantener un nivel de consistencia de los datos elevado en relación con el estado del sistema de archivos.

Respecto a la velocidad, el sistema de archivos ext3, aparte de permitir escribir datos más de una vez, en la mayoría de los casos tiene un rendimiento superior al que proporciona ext2 porque los "*journals*" de ext3 optimizan el movimiento de los cabezales de los discos duros. Se pueden seleccionar tres modos de *journaling* para optimizar la velocidad, pero, como contrapartida, la integridad de los datos se verá afectada.

Otra diferencia importante es que, existe fácil transición de la migración de ext2 a ext3 es muy sencilla y se pueden aprovechar las ventajas de un sólido sistema de archivos con *journaling* sin tener que volver a dar formato al sistema [9].

Espacio *swap*

El área de *swap* o espacio de intercambio, es utilizada para el intercambio de programas y procesos, se emplea cuando la memoria es limitada siendo un mecanismo para mover programas entre la memoria principal y la secundaria. Normalmente este espacio puede ser una participación dedicada, un archivo swap o una combinación de ambos, el tamaño no puede ser mayor de 2GB. La función del sistema operativo que es la encargada del intercambio de procesos entre disco y memoria se denominada *swapper* (intercambiador). La operación de escribir el programa en disco se conoce como *swap-out*, mientras que leer el programa de disco se denomina swap-in. La utilización del utilizar swap o *swapping*, consiste en alojar un proceso en disco hasta que esté listo para ejecutarlo nuevamente. El proceso de *swapping* presenta ventajas como:

- La gestión de procesos para controlar el grado de multiprogramación (planificación a medio plazo).
- Flexibilidad en la gestión de memoria, permitiendo una utilización más eficiente del espacio.

Para poder sacar un programa de la memoria en tiempo de ejecución, el *swapper* considera los siguientes criterios:

- El estado del proceso.
- La prioridad del proceso.
- El tamaño del programa.
- El tiempo que el programa lleva alojado en la memoria.

Sacar un proceso es dificultoso cuando se encuentra en estado de bloqueado por motivos de E/S. Para poder sacarlo hay dos alternativas:

- Impedir el swap-out de los procesos con operaciones DMA pendientes.
- La E/S se hace siempre sobre buffers del sistema (residentes).

Creación de una partición swap: Linux

1. Arrancar en modo rescate.
2. Crear una partición con parted o fdisk.
3. Configurar la partición con mkswap /dev/ hdbX
4. Activamos la partición swapon /dev/hdbX
5. Editamos /etc/fstab para el arranque: /dev/hdbX swap swap defaults 0 0
6. Para asegurar que este activa ingresamos: cat/proc/swaps o free

Eliminación de una partición swap: Linux

1. Se establece el modo de arranque en forma de rescate.
2. Se desactiva la partición swap: swapoff /dev/hdbX.
3. Eliminamos la entrada en /etc/fstab.
4. Eliminamos la partición con parted o fdisk.

Creación de un archivo swap: Linux

1. Escribir el comando: dd if=/dev/zero of=/swapfile bs=1024 count=tamañodeseado.
2. Configurar el archivo swap: mkswap /swapfile
3. Activar el archivo swap: swapon/swapfile
4. Editamos /etc/fstab para el arranque: /swapfile swap swap defaults 0 0
5. Confirmamos si está activado: cat /proc/swaps

Eliminación del archivo swap: Linux

1. Desactivar el archivo swap: swapoff /swapfile
2. Eliminamos la entrada en /etc/fstab
3. Eliminamos el archivo rm /swapfile

Volúmenes lógicos

Los volúmenes lógicos son técnicas de gestión de almacenamiento disponibles a partir de la versión 2.4 del núcleo de Linux heredadas del sistema operativo AIX, el dialecto Unix de IBM que permiten redimensionar las particiones y distribuirlas en varios discos.

En algunas distribuciones de Linux puede existir la restricción impuesta por el Gestor de Volúmenes Lógicos (LVM) de que el directorio /boot deba encontrarse en una partición real y no formar parte de ningún volumen lógico. Es obligatorio definir los volúmenes lógicos en el proceso de instalación cuando estos vayan a almacenar sistemas de archivos propios del sistema. El LVM consta de 3 elementos fundamentales [10]:

El instalador del sistema debe seguir los pasos de la Figura 14:

Volumen físico:	estructura que representa a un disco físico.
Volumen lógico:	estructura equivalente a un sistema de archivos Linux.
Grupo de volúmenes:	conjunto de varios volúmenes lógicos que pueden almacenarse en varios volúmenes físicos. Así, un disco puede contener varios sistemas de archivos y un sistema de archivos puede estar grabado en varios discos.

Figura 14. Elementos de un LVM

- Si la distribución GNU/Linux es antigua, crear una partición normal de tipo ext3 para el directorio /boot, ya sea incluido en el directorio raíz o en una partición propia.
- Definir un volumen físico en cada disco.
- Crear los grupos de volúmenes conjuntando adecuadamente los volúmenes físicos.
- Definir los volúmenes lógicos de cada grupo de volúmenes, asignando para cada uno de ellos su tamaño inicial y su punto de montaje.

Es recomendable dejar algún espacio sin asignar para poder ampliar las particiones que lo necesiten. El sistema incluye una gran variedad de mandatos para gestionar cada uno de los componentes del LVM. La Figura 15 describe la mayoría de estas instrucciones según su función [10].

Operación	Volúmen físico	Volúmen lógico	Grupo Volúmen
Crear	pvcreate	vgcreate	lvcreate
Eliminar	pvremove	vgremove	lvremove
Comprobar estado	pvscan	vgscan	lvscan
Ampliar tamaño		vgextend	lvextend
Reducir tamaño		vgreduce	lvreduce
Mostrar información	pvs	vgs	lvs
Mostrar atributos	pvdisplay	vgdisplay	lvdisplay

Figura 15. Comandos de configuración de LVM

En la Figura 16 muestra un ejemplo real usado para definir un grupo de volúmenes vg0 con 2 discos que usan un controlador RAID por hardware, en donde se definirán 3 volúmenes lógicos; posteriormente, el administrador podrá usar cada uno de ellos para montar los sistemas de archivos del servidor [10].

```
# pvcreate /dev/cciss/c1d0 /dev/cciss/c1d1
# pvs
  PV               VG   Fmt  Attr PSize   PFree
  /dev/cciss/c1d0      lvm2 --     1,36T   1,36T
  /dev/cciss/c1d1      lvm2 --   698,56G 698,56G
# vgcreate vg0 /dev/cciss/c1d0 /dev/cciss/c1d1
# vgs
  VG   #PV #LV #SN Attr   VSize VFree
  vg0    2   0   0 wz--n- 2,05T 2,05T
# lvcreate -L 300G vg0
# lvcreate -L 700G vg0
# lvcreate -L 700G vg0
# lvs
  LV    VG   Attr   LSize   Origin Snap%  Move Log Copy%
  lvol0 vg0  -wi-a- 300,00G
  lvol1 vg0  -wi-a- 700,00G
  lvol2 vg0  -wi-a- 700,00G
# vgs
  VG   #PV #LV #SN Attr   VSize VFree
  vg0    2   3   0 wz--n- 2,05T 395,69G
```

Figura 16. Ejemplo de configuración de LVM

La Figura 17 muestra la ejecución de la herramienta grafica systemconfig-lvm de CentOS 6.5, ejecutada en un servidor con un grupo de volúmenes formado por dos volúmenes físicos y tres lógicos.

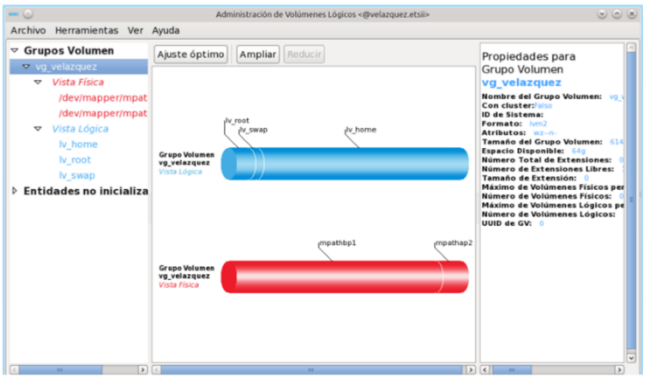

Figura 17. Ejemplo gráfico de configuración LVM

Conclusiones

Comprender los sistemas de archivos que utiliza un sistema operativo es importante, porque algunos de ellos no son compatibles con otro sistema operativo, por ejemplo, Windows

no es compatible con el sistema de archivos de Linux, pero en algunas distribuciones de Linux es posible montar un volumen como un disco adicional.

Existen sistemas de archivos que proporcionan integridad de datos al producirse un cierre no limpio del sistema, cuando el equipo se apaga o reinicia sin haber cerrado adecuadamente el archivo, un ejemplo de este sistema de archivos es ext3 de Linux porque por defecto los volúmenes son guardados para mantener un nivel de consistencia.

Referencias

[1] I. Lasso. Proyecto autodidacta. urlhttp://www.proyectoautodidacta.com/comics/quees-un-sistema-de-archivos/, 2015. Ultimo acceso: 17 Julio 2018.

[2] D. L. Castellanos. Sistemas operativos: Una guía de estudios, 2014.

[3] M. d. P. A. Ramos. Sistemas operativos monopuesto, 2000.

[4] G. L. Sergio. Migración y evaluación de un sistema de e/s paralela para plataformas Windows, 2009.

[5] V. A. Alberto. IOPFS: Sistema de ficheros basado en servidores intermedios de almacenamiento, 2011.

[6] J. Nio. Tareas básicas II (sistemas operativos monopuesto), 2000.

[7] A. S. TanenBaum. Sistemas operativos modernos tercera edición, 2009.

[8] M. C. Huguet. Administración de sistemas operativos en red, 2000.

[9] MitEdu. Proyecto autodidacta. urlhttp://web.mit.edu, 12 02 2000. [En línea]. Available: http://web.mit.edu/rhel-doc/3/rhel-sag-es-3/chext3.html, 2000. Ultimo acceso: 12/07/2018

[10] Ramón M Gómez Labrador. 14127 administración de servidores Linux (ubuntu/fedora/centos), 2014.