Sistemas de archivos distribuidos

BERTOK KLAUS, Universidad Central de Venezuela, Venezuela

Si consideramos que el compartir datos es fundamental, no sorprende que los sistemas de archivos compartidos constituyan el fundamento de muchas aplicaciones. Los sistemas de archivo distribuídos permiten que multiples procesos compartan datos durante largos períodos en forma segura y confiable. En este documento se intenta expresar con palabras sencillas el funcionamiento de NFS y abordar aspectos de su operatividad, así como de su arquitectura, seguridad y administración, en forma introductoria, como un caso de estudio clásico en el estudio de los sistemas de archivos ditribuídos, que puede servir para tener una visión general a la hora de ahondar en este tema. Tambien se comparan las diferentes versiones de NFS.

CCS Concepts: • **Computer systems organization → Distributed systems**; *Redundancy*; Replication; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Bertok Klaus. 2019. Sistemas de archivos distribuidos. *ACM Trans. Graph.* 37, 4, Article 111 (August 2019), 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCCIÓN

Uno de los objetivos clave de los sistemas distribuídos es compartir recursos, compartir información almacenada es quizas el aspecto más importante de la compartición de recursos distribuídos. Los sistemas de archivos distribuídos soportan la compartición de información en forma de archivos a través de Internet. Un sistema de archivos distribuído permite a los programas almacenar y acceder a archivos remotos del mismo modo que si fueran locales, permitiendo a los usuarios que accedan desde cualquier computador en una Intranet.

Otros servicios como el servicio de nombres, el servicio de autenticación de usuarios y el servicio de impresión pueden ser implementados más fácilmente, gracias a los sistemas de archivos que satisfacen sus necesidades de almacenamiento permanente. Los servidores web dependen de los sistemas de archivos para el almacenamiento de las páginas web que sirven.

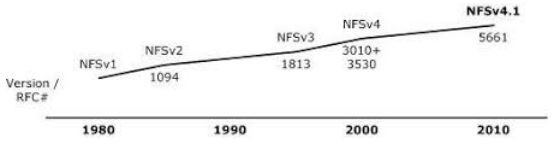

Fig. 1. Sistema de archivos de red y Linux - IBM Developer

Author's address: Bertok Klaus, Universidad Central de Venezuela, Los Chaguaramos, Caracas, Distrito Capital, Venezuela.

0730-0301/2019/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

El primer sistema de archivos de red se llamó File Access Listener y fué desarrollado en 1976 por la Digital Equipment Corporation (DEC). Una implementación de (DAP) Protocolo de acceso a datos, que era parte de la suite DECnet de protocolos. NFS (ver fig 1) fué el primer sistema de archivos de red moderno (construído sobre el protocolo IP), comenzó como un sistema de archivos experimentales desarrollado internamente en Sun en los primeros años de la década de los 80. NFS ha sido desarrollado durante los últimos 30 años, y representa un sistema de archivos extremadamente estable (y portable) que es escalable, de alto rendimiento y de calidad empresarial.

2 REQUISITOS DE UN SISTEMA DE ARCHIVOS DISTRIBUÍDOS

Muchos de los requisitos y potenciales obstáculos en el diseño de servicios distribuídos fueron ya observados en los primeros desarrollos de sistemas de archivos distribuídos. Inicialmente ofrecían transparencia de acceso y transparencia de ubicación, los requisitos de prestaciones, escalabilidad, control de concurrencia, tolerancia a fallos y seguridad surgieron y se fueron satisfaciendo en fases posteriores del desarrollo, a continuación se describen dichos requerimientos.

Transparencia El servicio de archivos es el servicio más fuertemente cargado de una intranet, por lo que su funcionalidad y desempeño son críticos. El diseño de un servicio de archivos debe soportar muchos de los requisitos de transparencia para sistemas distribuídos, tales como *transparencia de acceso*, ya que los programas de cliente no deben preocuparse de la distribución de los archivos.

Transparencia de ubicación: Los programas del cliente deben ver un espacio de nombres de archivos uniforme, los archivos o grupos de archivos pueden ser reubicados sin cambiar sus nombres de ruta, y los programas de usuario verán el mismo espacio de nombres en cualquier parte que sean ejecutados.
Transparencia de movilidad: Ni los programas del cliente ni las tablas de adminsitración de sistema en los nodos cliente necesitan ser cambiados cuando se mueven los archivos. Esta movilidad de archivos permite que archivos o, más comunmente conjuntos o volúmenes de archivos puedan ser movidos ya sea por los administradores del sistema o automáticamente.
Transparencia de prestaciones: Los programas del cliente deben continuar funcionando satisfactoriamente mientras la carga en el servicio varíe dentro de un rango especificado.
Transparencia de escala: El servicio puede ser aumentado por un crecimiento incremental para tratar con un amplio rango de cargas y tamaños de redes.
Actualizaciones concurrentes de archivos: Los cambios en un archivo por un cliente no deben interferir con la operación de otros clientes que acceden o cambian simultáneamente el mismo archivo.
Replicación de archivos: En un servicio de archivo que soporta replicación, un archivo puede estar representado por varias copias de su contenido en diferentes ubicaciones. Muy pocos servicios de

archivos soportan totalmente la replicación pero la mayoría soporta la caché local de archivos o porciones de archivos, una forma limitada de replicación.

Heterogeneidad del hardware y del sistema operativo: Las interfaces del servicio deben estar definidas de modo que el software del cliente y el servidor pueden estar implementados por diferentes sistemas operativos y computadores.

Tolerancia a fallos: El papel central de un servicio de archivos en los sistemas distribuídos hace que sea esencial que el servicio continúe funcionando aún en el caso de fallos del cliente y del servidor.

Consistencia: Cuando los archivos están replicados o en la caché en diferentes lugares hay un retardo inevitable en la propagación de las modificaciones hechas en un lugar hacia los otros lugares que mantienen copias, y esto puede producir alguna desviación de la semántica de una copia.

Seguridad: Virtualmente todos los sistemas de archivos proporcionan mecanismos de control de acceso basado en el uso de listas de control de acceso. En sistemas de archivos distribuídos, hay una necesidad de autenticar las solicitudes del cliente por lo que el control de acceso en el servidor está basado en identificar al usuario correcto y proteger el contenido de los mensajes de solicitud y respuesta con firmas digitales y opcionalmente encriptación de datos secretos.

Eficiencia: Un servicio de archivos distribuídos debe ofrecer posibilidades con la misma potencia y generalidad que las que se encuentran en los sistemas de archivos convencionales y deben de proporcionar un nivel de prestaciones comparable.

3 ARQUITECTURA NFS

NFS sigue el modelo cliente-servidor. En las siguientes líneas se intenta describir de una forma sencilla, los distintos componentes de la arquitectura de NFS, y para ello nos centramos en las figuras 2 y 3. El servidor implementa el sistema de archivos compartido y de almacenamiento al cual el cliente se adjunta. El cliente implementa la interfaz de usuario al sistema de archivos compartido, montado dentro del espacio local de archivos del cliente.

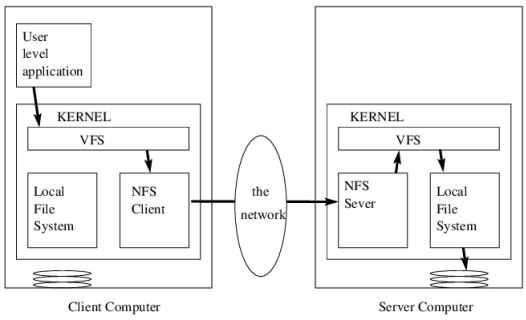

Fig. 2. Arquitectura NFS, researchgate.net

Dentro de Linux, el sistema de archivos virtuales (VFS) provee el medio que soporta multiples sistemas de archivos concurrentemente en un nodo (tales como ISO-9660 en un CD-ROM o ext3fs en un disco local). El VFS determina que almacenamiento está destinado para una solicitud, por consiguiente determina que sistema de archivos necesitará dicho requerimiento, por esa razón NFS es un sistema de archivos que se puede enchufar o agregar como cualquier otro, la única diferencia con NFS es que los requerimeintos de I/O no pueden ser satisfechos localmente, por lo que deben atravesar la red para completarse.

El servicio NFS utiliza, para su funcionamiento, los servicios proporcionados por el protocolo de representación de datos externos (eXternal Data Representation, XDR) de la capa de presentación del modelo OSI y el protocolo de llamada a procedimiento remoto (Remote Procedure Call, RPC) de la capa de sesión del modelo OSI. Por debajo, dependiendo de la versión del protocolo, se utiliza UDP y/o TCP para la capa de transporte e IP para la capa de red. En concreto, la versión 1 de NFS utiliza solo UDP, mientras que las versiones 2 y 3 de NFS utilizan TCP y UDP para la capa de transporte, mientras que la versión 4 solamente utiliza TCP.

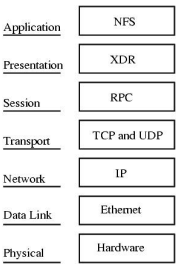

Fig. 3. NFS en capa 7 del modelo OSI, rutgers.edu

Una vez que se determina que un requerimiento debe ser atendido por NFS, VFS le pasa este a una instancia de NFS, dentro del kernel. NFS interpreta el requerimiento de I/O y la traduce a un procedimiento NFS (OPEN, ACCESS, CREATE, READ, CLOSE, REMOVE y así sucesivamente). Esos procedimientos, que están documentados en el respectivo RFC de NFS, especifíca el comportamiento dentro del protocolo NFS. A grandes rasgos, un RFC es un documento que especifica un protocolo de Internet.

Una vez hecho el requerimiento de I/O se selecciona un procedimiento a ejecutar, este procedimiento se lleva a cabo en la capa de (RPC), es decir llamadas a procedimientos entre sistemas, este empaca el requerimiento y sus argumentos y los envía al nodo remoto, maneja y le hace seguimiento a la respuesta proporcionandosela al solicitante del requerimeinto.

Adicionalmente (ver figura 3) RPC incluye una capa de interoperabilidad llamada external data representation (XDR).Como el nombre indica, el cual asegura que todos las particpantes NFS hablen el mismo lenguaje cuando se trata de tipos de datos. Recordemos que NFS se puede implementar en diferentes arquitecturas, cuando una arquietectura específica lleva a cabo un requerimiento, la representación de los tipos de datos puede diferir entre quien hace el requerimiento y quien lo atiende.

XDR cuida realizar la conversión de los tipos a una representación común de tal forma que todas las arquitecturas puedan interoperar y compartir sistemas de archivos. XDR especifica el formato de bit para tipos tales como float y el orden de byte para tipos de datos tales como arreglos de longitud fija y variable. Otro aspecto importante

es que XDR es ampliamente conocido en NFS, sin embargo es una especificación muy útil en cualquier aplicación multi-arquitectura.

Una vez que XDR ha traducido los datos a una representación en común el requerimiento se transfiere a través de la red dado un protocolo en la capa de transporte. En sus inicios NFS usó UDP pero hoy en día se usa TCP para mayor confiabilidad.

A nivel del servidor de NFS, este opera de la misma forma, el requerimiento fluye a través del stack de red a través de RPC/XDR para traducir los tipos de datos a la arquitectura del servidor. El servidor pasa el requerimiento al demonio (proceso en background o segundo plano) el cual identifica el arbol del sistema de archivos objetivo que se requiere para atender el requerimiento, ya que el servidor NFS puede exportar distintos sistemas de archivos, y VFS es de nuevo utilizado para obtener el sistema de archivos en el almacenamiento local. El sistema de archivos local en el servidor puede ser un sistema de archivos típico tal como ext4. De tal forma que NFS no es un un sistema de archivos en un sentido tradicional sino un protocolo para acceder sistemas de archivos remotamente. De Acuerdo a la figura 1, NFS ha evolucionado hasta llegar a la versión 4.1, Para redes de alta latencia, NFSv4 implementa lo que se ha llamado el procedimiento compuesto. Este procedimiento esencialmente permite que multiples llamadas RPC sean empaquetadas dentro de un único requerimiento para minimizar la tasa de transferencia sobre la red.

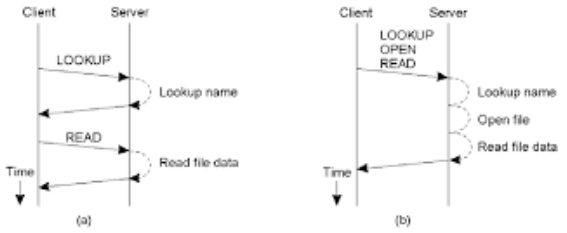

Fig. 4. NFS procedimiento compuesto. si.deis.unical.it

Desde la perspectiva del cliente, la primera operación que ocurre dentro de NFS se llama mount. Mount representa el montaje de un sistema de archivos remoto en el espacio del sistema de archivos local. Este proceso comienza como una llamada a mount (llamada al sistema en linux), la cual es enrutada a través de VFS al componente NFS. Después de establecer el número del puerto para el montaje a través de un requerimiento RPC get_port al servidor remoto, el cliente ejecuta un requerimiento de montaje RPC. Este requerimiento ocurre entre el cliente y un demonio (proceso en background) especial llamado rpc.mountd.

Este demonio chequea el requerimiento del cliente en la lista actual de sistemas de archivos exportados, si el sistema de archivos requerido existe y el cliente tiene acceso, una respuesta mount de tipo RPC, establece el descriptor de archivo para el sistema de archivos. El cliente almacena la información del montaje remoto con el punto de montaje local y establece la posibilidad de ejecutar requerimientos de I/O. Este protocolo representa un aspecto de seguridad potencial, por consiguiente NFSv4 reeemplaza este protocolo de montaje con llamadas a RPC para manejar el punto de montaje.

4 SEGURIDAD EN LOS SISTEMAS DE ARCHIVOS DISTRIBUÍDOS

Para ilustrar porqué es importante la seguridad en los sistemas de archivos distribuídos, en la siguiente tabla se muestra el conjunto de comandos en una máquina linux que explota una configuración débil de un servidor nfs. La técnica consiste en aprovechar que el servidor exporta /home y que al insertar claves ssh en el .ssh del directorio home de un usuario se puede ingresar vía ssh a la máquina remota como super usuario y acceder no solo a /home sino a toda la máquina remota.

Número	Comando
1.	nmap -sV 192.168.100.25
2.	showmount -e 192.168.100.25
3.	mkdir /tmp/nfs
4.	mount -t nfs 192.168.100.25:/home/tmp/nfs
5.	ls -l
6.	cd /home/tmp/nfs/aperez/.ssh
7.	ls -la /home/tmp/nfs/aperez/.ssh
8.	ssh-keygen
9.	echo <salida del comando anterior> » authorized_keys
10.	ssh -i aperez_rsa aperez192.168.100.25

Table 1. Conjunto de comandos que muestran un exploit a un nfs débilmente configurado, resources.infosecinstitute.com

NFS trabaja muy bien compartiendo sistemas de archivos enteros con un gran número de hosts conocidos de una manera muy transparente. Sin embargo, esta facilidad de uso trae una variedad de problemas potenciales de seguridad. En NFS2 o NFS3, NFS controla quien puede montar y exportar sistemas de archivos, basados en la máquina que hace la petición, no el usuario que utiliza el sistema de archivos, los hosts tienen que tener los derechos para montar los sistemas de archivos exportados explícitamente. El control de acceso no es posible para usuarios, aparte de los permisos de archivos y directorios. En otras palabras, una vez que un sistema de archivos es exportado vía NFS, cualquier usuario en cualquier máquina remota conectada al servidor NFS puede acceder a los datos compartidos. Para limitar estos riesgos potenciales, los administradores sólo pueden permitir acceso de sólo-lectura o reducir a los usuarios a un usuario común y groupid. Pero estas soluciones pueden impedir que la compartición NFS sea usada de la forma en que originalmente se pensó.

Adicionalmente, si un atacante gana el control del servidor DNS usado por el sistema que exporta el sistema de archivos NFS, el sistema asociado con un nombre de host concreto o nombre de dominio totalmente cualificado, puede ser dirigido a una máquina sin autorización. En este punto, la máquina desautorizada es el sistema que tiene permitido montar la compartición NFS, ya que no hay intercambio de información de nombre de usuario o contraseña para proporcional seguridad adicional al montaje NFS.

Los comodines o metacaracteres deben ser usados lo menos posible cuando garantizamos el acceso a una compartición NFS. El uso de los comodines puede incluir más sistemas de los que se desean.

También es posible restringir el acceso al servicio portmap a través de los TCP wrappers. El acceso a los puertos usados por portmap, rpc.mountd y rpc.nfsd se puede limitar creando reglas de cortafuegos o firewall con iptables (firewall que forma parte del kernel de linux).

EN NFS4, muchas cosas cambiaron, El lanzamiento de NFSv4 trajo consigo una revolución para la autentificación y la seguridad cuando se comparten directorios a través de NFS. Con NFS4, los mecanismos de seguridad obligatorios están orientados hacia la autenticación de usuarios individuales y no a máquinas clientes, como lo hace NFS2 y NFS3. NFS4 incluye el soporte a ACL (Listas de control de acceso) basado en el modelo de Microsoft Windows NT, no en el modelo POSIX (Interfaz Portátil de Sistema Operativo para Unix).(Es un estándar del IEEE que define un conjunto de servicios del sistema operativo.) por sus funcionalidades y porque es implementado ampliamente. NFSv2 y NFSv3 no son compatibles con los atributos nativos de ACL.

Una vez que el sistema de archivos es montado como lectura / escritura por un host remoto, la única protección que tiene cada archivo compartido son sus permisos. Si dos usuarios que comparten el mismo valor de identificador de usuario montan el mismo NFS, ellos podrán modificar sus archivos mutuamente. Adicionalmente, cualquiera con acceso root (superusuario) en el sistema cliente puede usar el comando su - para volverse un usuario que tenga acceso a determinados archivos a través de la compartición NFS.

Otra funcionalidad de seguridad importante de NFSv4 es la eliminación del demonio rpc.mountd. El demonio rpc.mountd presentaba posibles agujeros de seguridad debido a la forma en que trataba con los manejadores de archivos.

5 ADMINISTRACIÓN DE LOS SERVICIOS PROVISTOS POR UN SISTEMA DE ARCHIVOS DISTRIBUÍDOS

Siguiendo con NFS, los elementos a tomar en cuenta, en su configuración y administración, los archivos y los demonios (procesos en background) serían los siguientes:

- /etc/exports.
- /etc/hosts.allow.
- /etc/hosts.deny.

Únicamente se necesita /etc/exports para poner a punto NFS, pero para agregarle seguridad a la hora de compartir un sistema de archivos se usan los otros dos. Cuando un servidor NFS recibe un requerimiento de un cliente, el primero chequea el hosts.allow despues hosts.deny y si no hay ninguna restricción entonces se permite el acceso.

/etc/exports Es el standard para controlar cual sistema de archivos será exportado a cual o cuales máquinas, tambien como la especificación de opciones particulares que controlan cada aspecto de NFS.

Cada entrada en /etc/exports indíca un volúmen a compartir y como este es compartido. Las siguientes dos líneas son un ejemplo de ello.

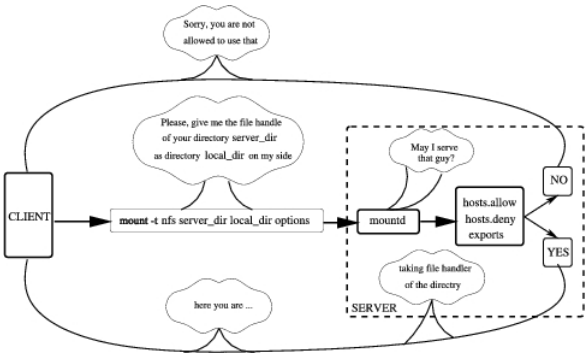

Fig. 5. Compartición de archivos con NFS linuxjournal.com

vi /etc/exports
/usr/share/doc cliente01(rw) cliente02(ro)
Las opciones (ro) read only (solo lectura) o (rw) (read write) (lectura y escritura) deben estar justo a continuación del nombre de la máquina a la que se le dá el permiso. Si no se hace eso y se coloca la siguiente entrada:
/usr/share/doc cliente01 (rw)
compartirá /usr/share/doc a cliente01 con las opciones por default, y cualquier host puede tener acceso a /usr/share/doc con las opciones de lectura y escritura. Por esa razón es muy importante verificar la sintaxis.
Una opción importante es: ***no_root_squash*** esta opción mapea el usuario y grupo root de el cliente NFS a las cuentas locales de root y group de la máquina local, por lo tanto es un punto de seguridad que debe ser cuidado, por lo general se usa la opción: ***root_squash***, en este caso el usuario root de los clientes se mapea a nobody, así un cliente no puede dejar malware para que otro lo ejecute. Así como el setuid bit.

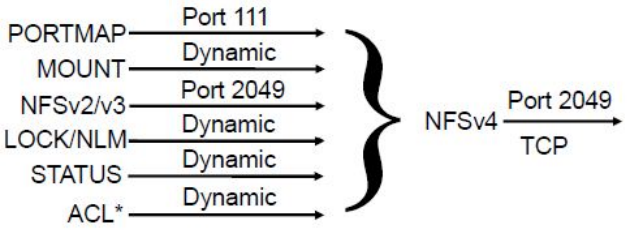

Fig. 6. NFS4 consolidación de puertos, storagegaga.wordpress.com

Una de las diferencias entre NFS3 y NFS4 es que consolida la mayoría de los servicios TCP/IP en puertos bien conocidos los cuales se pueden definir con seguridad en el firewall (cortafuegos) ver figuras 6 y 7.
Demonios, inetd y portmapper.
Frecuentemente los servicios son llevados a cabo por los llamados demonios. Un demonio es un programa que abre un determinado puerto, y espera a recibir peticiones de conexión. Si se recibe una petición de conexion lanza un proceso hijo que aceptará la conexión, mientras el padre continua escuchando a la espera de más peticiones.

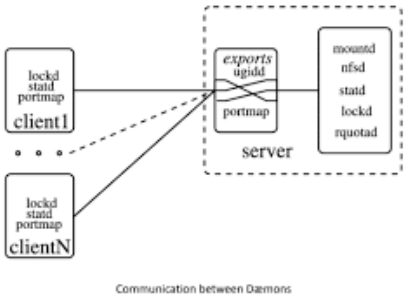

Fig. 7. Servidor NFS en Linux, knowitgeeks.com

Este concepto tiene el inconveniente de que para cada servicio ofrecido, se necesita escuchar un demonio que escuche las conexiones a un puerto. Por ello se ejecuta una especie de super servidor que crea sockets para varios servicios y escucha en todos ellos simultáneamente llamando a select (colas de sockets asíncronos), cuando un nodo remoto requiere uno de los servcios, el super-servidor lo recibe y llama al servidor especificado para ese puerto. El super servidor más usado es inetd. Ya se vió que tanto los RPC como NFS fueron desarrollados por SUN, y que NFS usa los RPC para lograr su funcionalidad. Sin embargo los RPC no usan al superdemonio inetd sino el portmapper, cabe destacar que uno de los servicios más complejos que usan RPC es NFS:

```
$ rpcinfo -p program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
100003 4 udp 2049 nfs
100003 2 tcp 2049 nfs
100003 3 tcp 2049 nfs
100003 4 tcp 2049 nfs
100024 1 udp 32770 status
100021 1 udp 32770 nlockmgr
```

Como puede observarse del listado anterior, portmapper opera en forma semejante a un servicio atendido por inetd, pero este en esencia controla la tabla de mapeo de relación de programas RPC y números de versiones locales de TCP o UDP de números de puertos, como cual es el servicio y su disponibilidad, el portmapper ocupa el puerto 111 tanto en UDP como en TCP.
Como se describió en los párrafos anteriores, El acceso al servidor NFS puede ser controlado con los archivos /etc/host.allow, /etc/host.deny y /etc/exports. rpc.nfsd recibe los requerimientos NFS desde un sistema remoto, rpc.mountd ejecuta requerimientos de montaje y desmontaje, rpc.portmapper mapea requerimientos remotos al demonio NFS apropiado.
rpc.rquotad provee manejo de quota de discos de usuario. rpc.statd proveé bloqueo de servicios cuando el host remoto rebootea.
Actualmente hay tres versiones de NFS , la version 2, es la más vieja y la más ampliamente soportada, la version 3 tiene más caracteristicas, incluyendo descriptores de archivo de 64 bits, escrituras asíncronas seguras, y un manejo de errores mas robusto. NFSv4 trabaja a través de firewalls y sobre internet, no requiere más al portmapper, soporta ACLs y usa operaciones full estado.

6 RESÚMEN

En este breve artículo se ha descrito en forma sencilla y muy general, los elementos que hay que considerar en el diseño de un sistema de archivos distribuído, se ha escogido a NFS, como un caso concreto de un sistema de archivos distribuído con al menos tres décadas de desarrollo y mejoras, el funcionamiento de NFS, sus origenes, evolución, arquitectura y aspectos de su administración.
También se ha visto, que NFS ha sido diseñado bajo el paradigma cliente / Servidor, que utiliza una serie de capas y protocolos para poderlo implementar, que hay una serie de demonios que cumplen una función específica y que hay aspectos en su seguridad que deben ser manejados cuidadosamente para no poner en riesgo los datos exportados.
A continuación se resúmen, las capas, procesos en backgroud, archivos de seguridad y comandos más frecuentemente usados en NFS, que se han visto en los párrafos anteriores, excepto autofs y nfsstat, el primero permite montar los servicios de NFS por demanda y el segundo permite obtener estadísticas de NFS.

Capas de NFS: llamadas al sistema, RPC, XDR, TCP, VFS.
Demonios: mountd, lockd, quotad, nfsd.
Seguridad: host.allow, host.deny, exports, fstab.
Comandos: rpcinfo -p, showmount -e, mount, umount, nfsstat -s, mkdir, chmod.
Servicios: : nfs-kernel-server, autofs.

REFERENCES

[1] Nils Magnus: Ghost on the Loose, http://www.linux-magazine.com/Issues/2009/99/Surviving-NFS-3
[2] M. Jones: Network file systems and Linux, https://developer.ibm.com/tutorials/l-network-filesystems/
[3] Linux for eenginering and It aplications, http://coewww.rutgers.edu/www1/linuxclass2007/lessons/lesson5/OSI.html
[4] Uploaded by Alex Brodsky, https://www.researchgate.net/figure/NFS-software-architecture
[5] Starting share files with NFS by Olexiy Tykhomyrov, https://www.linuxjournal.com/article/4880
[6] NFS (Network File System) Server in Linux, https://knowitgeeks.com/nfs-server-linux/
[7] NFS (Network File System) version 4, http://storagegaga.com/tag/kerberos/
[8] NFS (Network File System) version 4, http://estigia.fi-b.unam.mx/chontalpa/pdf/prared13.pdf
[9] NFS (Sistema de archivos de red (NFS), https://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/s1-nfs-security.html
[10] Sistemas distribuídos conceptos y diseño, George coulouris, JeanDollimore, Tim Kindberg
[11] Sistemas distribuídos principios y paradigmas, Andrew S Tanembaum

A METODOS DE INVESTIGACIÓN

A.1 Parte Uno

En general, hay mucha información diseminada por internet, y cada artículo, blog, fuente de información, enfoca a NFS desde una perspectiva diferente, seguridad, administración, funcionamiento,

diseño, por lo que es necesario consolidar de manera consistente toda esa información para poder generar un resúmen muy general de algunas de sus características más importantes, todo ello basado en los lineamientos del libro de texto utilizado, el Tanembaum.