Universidad Nacional Autónoma de México

Facultad de Ingeniería

Sistemas Operativos

Grupo #06

Sistemas de Archivos Distribuidos

Miyasaki Sato Yuichi Vicente

No. Cuenta 318586465

Fecha de entrega: 12/10/24

Profesor: Ing. Gunnar Eyal Wolf Iszaevich

Índice

1.- Introducción.

1.1.- Definición.

Un sistema de archivos distribuido o también conocido como DFS (distributed file system) es un método para almacenar y acceder a archivos basado en una arquitectura cliente - servidor, en un sistema de archivos distribuido, uno o más servidores centrales almacenan archivos que pueden ser accedidos, con los derechos de autorización adecuados, por cualquier número de clientes remotos en la red.

También un sistema de archivos distribuido gestiona, organiza, almacena, protege, recupera y comparte archivos de datos, por lo que las aplicaciones y los usuarios pueden almacenar o acceder a archivos de datos en el sistema de la misma manera que lo harían con un archivo local [1].

1.2.- Características principales.

- **Transparencia de acceso:** Los usuarios pueden acceder a los archivos como si estuvieran almacenados directamente en sus dispositivos locales.
- **Transparencia de la ubicación:** Las máquinas host no requieren conocer la ubicación de los datos del archivo, ya que el DFS se encarga de gestionarlo.
- **Actualizaciones concurrentes de archivos:** Los cambios en un archivo realizados por un cliente no deberían interferir con la operación de otros clientes que acceden o modifican simultáneamente el mismo archivo.
- **Cifrado de datos:** El DFS protege los datos cifrándolos mientras se transmiten a través del sistema.
- **Heterogeneidad:** Las interfaces de servicio deben definirse de manera que el software cliente y servidor pueda implementarse en diferentes sistemas operativos y computadoras.
- **Replicación:** Un DFS también replicará conjuntos de datos en diferentes clústeres al copiar las mismas piezas de información en múltiples clústeres. Esto ayuda a lograr tolerancia a fallos, permitiendo recuperar los datos en caso de un fallo de nodo o clúster, así como una alta concurrencia, que permite que la misma pieza de datos sea procesada al mismo tiempo.
- **Compatibilidad con varios protocolos:** Los hosts pueden acceder a los archivos utilizando una variedad de protocolos, como Server Message Block (SMB), Network File System (NFS) y Portable Operating System Interface (POSIX), entre otros.

1.3.- Funcionamiento.

A través de DFS, se establece una red que conecta terminales y servidores, formando un sistema de archivos paralelo con un clúster de nodos de almacenamiento. Este sistema está estructurado bajo un espacio de nombres único y un conjunto de almacenamiento, lo que facilita el acceso rápido a los datos desde múltiples hosts o servidores al mismo tiempo [2].

Los datos pueden almacenarse en una variedad de dispositivos, que incluyen discos duros, unidades de estado sólido y almacenamiento en la nube pública. Independientemente de su ubicación, DFS puede configurarse como un espacio de nombres autónomo, que utiliza un solo servidor host, o como un espacio de nombres basado en dominios, que incorpora varios servidores host [2].

Cuando un usuario selecciona un nombre de archivo para acceder a los datos, DFS consulta diversos servidores en función de la ubicación del usuario y proporciona la primera copia disponible del archivo en ese grupo de servidores [2].

Este método evita la sobrecarga de cualquier servidor cuando múltiples usuarios acceden a los archivos y asegura que los datos sigan siendo accesibles, incluso si un servidor falla. Gracias a la función de replicación de archivos de DFS, cualquier modificación realizada en un archivo se refleja en todas las instancias de dicho archivo en los nodos del servidor [2].

1.4.- Ventajas y desventajas de usar DFS.

Ventajas:

Diseñado para grandes conjuntos de datos: DFS está diseñado para almacenar y gestionar archivos muy grandes, típicamente de gigabytes o terabytes de tamaño. Los sistemas de archivos tradicionales no son tan adecuados para esta tarea [3].

Tolerancia a fallos: DFS está diseñado para ser tolerante a fallos, lo que significa que puede seguir operando incluso si parte del sistema falla. Esto se logra a través de la replicación: cada archivo se almacena en múltiples nodos del sistema [3].

Escalabilidad: DFS puede escalarse fácilmente al agregar más nodos al sistema. Esto le permite crecer según sea necesario para soportar conjuntos de datos más grandes y más usuarios [3].

Simplicidad: DFS tiene un diseño simple que facilita su implementación y mantenimiento [3].

Desventajas:

Más lento que otros sistemas de archivos: DFS no es tan rápido como algunos otros sistemas de archivos, como los sistemas de archivos locales o NTFS (en Windows). Esto se debe en parte a su diseño: dado que los archivos se replican en múltiples nodos, deben leerse desde varias ubicaciones al acceder a ellos, lo que lleva más tiempo que leer desde una sola ubicación. Además, la compresión de datos no se utiliza por defecto en DFS, lo que afecta aún más el rendimiento [3].

No es adecuado para archivos pequeños: Debido a su diseño, DFS no es eficiente para manejar archivos pequeños [3].

2.- Arquitectura de un sistema de archivos distribuido.

Una arquitectura que proporciona una separación clara y eficiente de las principales responsabilidades en la gestión del acceso a archivos se puede lograr estructurando el servicio de archivos en tres componentes clave: un servicio de archivos planos, un servicio de directorios y un módulo cliente.

2.1 Servicio de archivos planos.

El servicio de archivos planos se ocupa para implementar operaciones sobre el contenido de los archivos. Se utilizan identificadores únicos de archivos (UFIDs) para referirse a los archivos en todas las solicitudes de operaciones del servicio de archivos planos. Cuando el servicio de archivos planos recibe una solicitud para crear un archivo, genera un nuevo UFID para el archivo y lo devuelve al solicitante.

En comparación con la interfaz de UNIX, nuestro servicio de archivos planos no tiene operaciones de apertura y cierre; los archivos pueden ser accedidos inmediatamente citando el UFID correspondiente. La interfaz de nuestro servicio de archivos planos difiere de la del sistema de archivos de UNIX principalmente por razones de tolerancia a fallos:

2.1.1 Operaciones repetibles: Con la excepción de la creación, las operaciones son idempotentes, es decir, la ejecución repetida de la operación produce el mismo resultado. La ejecución repetida de Crear genera un archivo nuevo diferente en cada llamada.

2.1.2 Servidores sin estado: La interfaz es adecuada para su implementación en servidores sin estado. Los servidores sin estado pueden reiniciarse después de un fallo y reanudar la operación sin necesidad de que los clientes o el servidor restauren ningún estado anterior.

2.1.3 Operaciones del servicio de archivos planos: estas son algunas de las operaciones que se utilizan:

Read(FileId, i, n) → Data — throws *BadPosition*	If $1 \le i \le Length(File)$: Reads a sequence of up to *n* items from a file starting at item *i* and returns it in *Data*.
Write(FileId, i, Data) — throws *BadPosition*	If $1 \le i \le Length(File)+1$: Writes a sequence of *Data* to a file, starting at item *i*, extending the file if necessary.
Create() → FileId	Creates a new file of length 0 and delivers a UFID for it.
Delete(FileId)	Removes the file from the file store.
GetAttributes(FileId) → Attr	Returns the file attributes for the file.
SetAttributes(FileId, Attr)	Sets the file attributes (only those attributes that are not shaded in Figure 12.3).

Imagen 1 *(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004).*

2.2 Servicio de directorios.

El servicio de directorios proporciona un mapeo entre los nombres de texto de los archivos y sus Identificadores Únicos de Archivos (UFIDs). Los clientes pueden obtener el UFID de un archivo proporcionando su nombre de texto al servicio de directorios [5].

En otras palabras, este servicio actúa como un intermediario que traduce los nombres legibles por humanos en identificadores únicos que el sistema utiliza internamente para gestionar y acceder a los archivos [5].

2.2.1 Operaciones del servicio de directorios: estas son algunas de las operaciones que podemos realizar con el servicio de directorios:

Lookup(Dir, Name) → FileId — throws *NotFound*	Locates the text name in the directory and returns the relevant UFID. If *Name* is not in the directory, throws an exception.
AddName(Dir, Name, FileId) — throws *NameDuplicate*	If *Name* is not in the directory, adds *(Name, File)* to the directory and updates the file's attribute record. If *Name* is already in the directory, throws an exception.
UnName(Dir, Name) — throws *NotFound*	If *Name* is in the directory, removes the entry containing *Name* from the directory. If *Name* is not in the directory, throws an exception.
GetNames(Dir, Pattern) → NameSeq	Returns all the text names in the directory that match the regular expression *Pattern*.

Imagen 2 *(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004).*

2.3 Modulo Cliente.

Un módulo cliente se ejecuta en cada computadora cliente, integrando y extendiendo las operaciones del servicio de archivos planos y el servicio de directorios bajo una única interfaz de programación de aplicaciones (API) disponible para los programas a nivel de usuario en las computadoras cliente [5].

2.3.1 Control de acceso.

Se realiza una verificación de acceso siempre que un nombre de archivo se convierte en un UFID.

Con cada solicitud del cliente se envía la identidad del usuario, y el servidor realiza comprobaciones de acceso para cada operación de archivo.

2.4 Arquitectura de un sistema de archivos distribuido completo.

La imagen muestra de manera completa la arquitectura del sistema de archivos distribuido, donde el módulo cliente en la computadora cliente interactúa con los servicios de directorios y archivos planos en la computadora servidora. Esta estructura permite a los programas de aplicación realizar operaciones de gestión de archivos y nombres de manera eficiente.

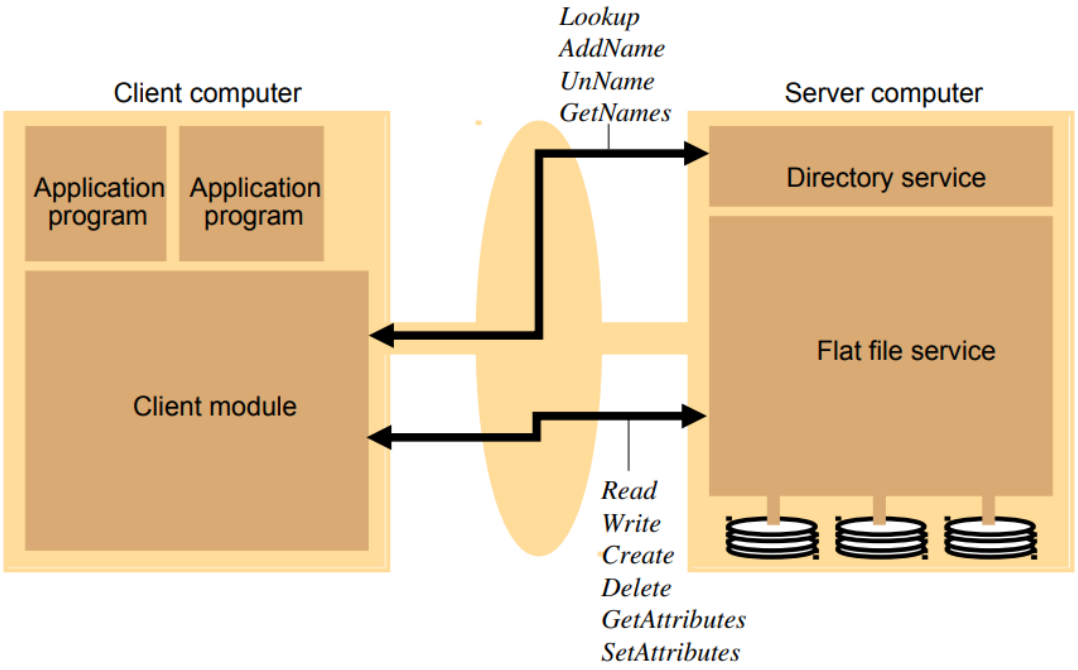


Imagen 3 *(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004).*

3.- Ejemplos de Sistemas de archivos distribuidos.

3.1 NFS (Network File System).

3.1.1 Descripción.

NFS es un protocolo de red que facilita a los sistemas operativos el acceso transparente a archivos y directorios ubicados en servidores remotos. Originalmente creado por Sun Microsystems en los años 80, se ha establecido como un estándar para el intercambio de archivos en redes.

3.1.2 Funcionalidad.

El módulo NFS se encuentra en el núcleo de cada computadora. Las solicitudes relacionadas con archivos en un sistema de archivos remoto son convertidas por el módulo del cliente en operaciones del protocolo NFS y luego enviadas al módulo del servidor NFS en la computadora que contiene el sistema de archivos correspondiente. Los módulos NFS del cliente y del servidor se comunican mediante llamadas a procedimientos remotos. La interfaz RPC del servidor NFS es abierta, o sea, que cualquier proceso puede enviar solicitudes a un servidor NFS; si estas solicitudes son válidas y contienen credenciales de usuario correctas, se ejecutarán.

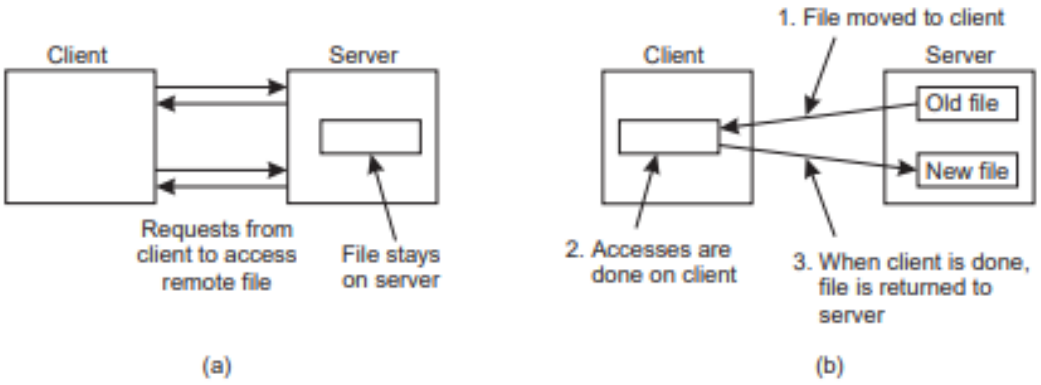


Imagen 4 *(Distributed Systems Distributed File System [2] Distributed File System, n.d.).*

3.1.3 Arquitectura.

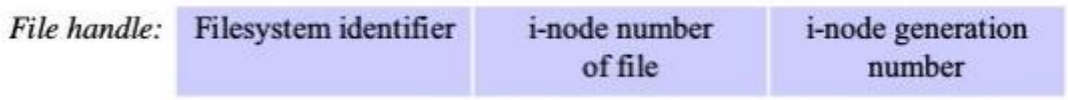
La arquitectura de NFS, aparte de los 3 componentes que ya habíamos visto con anterioridad, maneja varios componentes con diferentes funcionalidades, por ejemplo:

3.1.3.1 Interfaz del servidor NFS.

Las operaciones de archivos y directorios están integradas en un solo servicio; la creación e inserción de nombres de archivos en directorios se realiza mediante una única operación de creación, que toma como argumentos el nombre de texto del nuevo archivo y el manejador de archivos para el directorio de destino [4].

3.1.3.2 Sistema de Archivos Virtual (VFS).

La integración se logra mediante un módulo de sistema de archivos virtual (VFS), que se ha añadido al núcleo de UNIX para distinguir entre archivos locales y remotos. Los identificadores de archivos utilizados en NFS se llaman manejadores de archivos [4].



(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004)

El campo del identificador del sistema de archivos es un número único que se asigna a cada sistema de archivos cuando se crea [4].

El número del i-nodo es necesario para localizar el archivo en el sistema de archivos y también se utiliza para almacenar sus atributos; los números de i-nodo se reutilizan después de que se elimina un archivo [4].

El número de generación del i-nodo es necesario para incrementar cada vez que se reutilizan los números de i-nodo después de que se elimina un archivo [4].

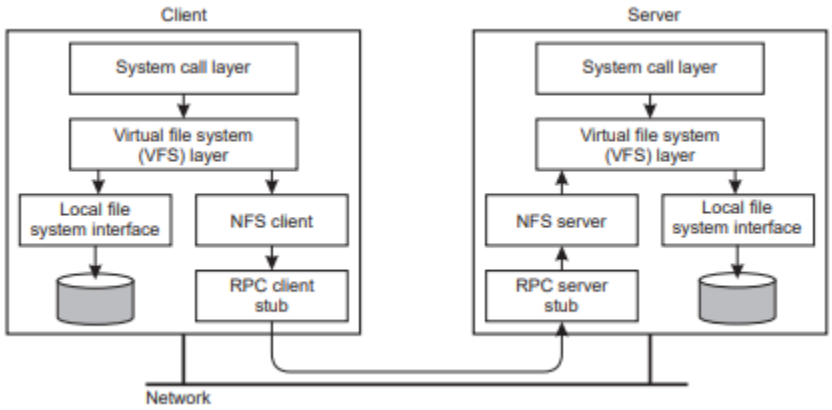
3.1.3.3 Integración del cliente.

El módulo cliente NFS coopera con el sistema de archivos virtual en cada máquina cliente. Funciona de manera similar al sistema de archivos UNIX convencional, transfiriendo bloques de archivos hacia y desde el servidor y almacenando en caché los bloques en la memoria local siempre que sea posible. Si el archivo es local, el v-nodo contiene una referencia al índice del archivo local. Si el archivo es remoto, contiene el manejador de archivos del archivo remoto [4].

3.1.3.4 Control de acceso y autenticación:

El servidor NFS es sin estado y no mantiene los archivos abiertos en nombre de sus clientes. Por lo tanto, el servidor debe verificar la identidad del usuario con respecto a los atributos de permisos de acceso del archivo en cada solicitud, para determinar si se permite al usuario acceder al archivo de la manera solicitada [4].

3.1.3.5 Esquema completo de la arquitectura NFS:



(*Distributed Systems Distributed File System [2] Distributed File System, n.d.)*

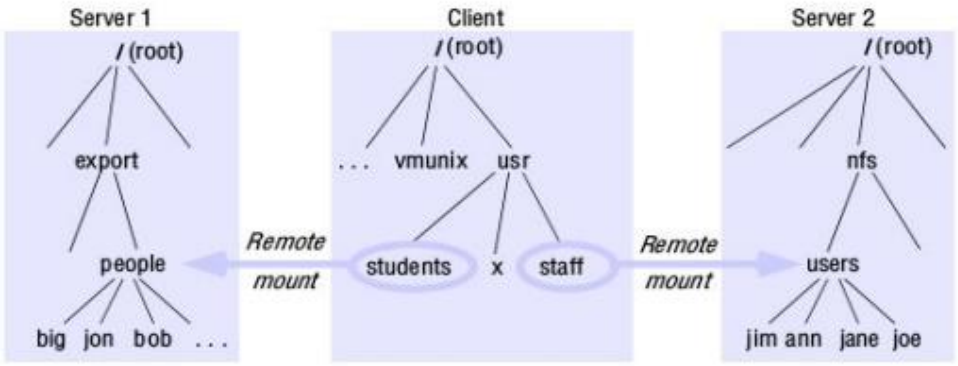
3.1.4 Características adicionales del sistema de archivos NFS.

3.1.4.1 Servicios de montaje.

Montar consiste en hacer accesible un grupo de archivos en una estructura de sistema de archivos a un usuario o grupo de usuarios [4].

Los sistemas de archivos remotos pueden ser montados de forma hard o soft en una computadora cliente. Cuando un proceso de nivel de usuario accede a un archivo en un sistema de archivos que está montado de forma hard, el proceso se suspende hasta que se complete la solicitud, y si el host remoto no está disponible por cualquier motivo, el módulo cliente NFS continúa intentando la solicitud hasta que se satisface [4].

Así, en el caso de una falla del servidor, los procesos de nivel de usuario se suspenden hasta que el servidor se reinicie y luego continúan como si no hubiera habido ninguna falla. Pero si el sistema de archivos relevante está montado de forma soft, el módulo cliente NFS devuelve una indicación de falla a los procesos de nivel de usuario después de un pequeño número de reintentos [4].



(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004).

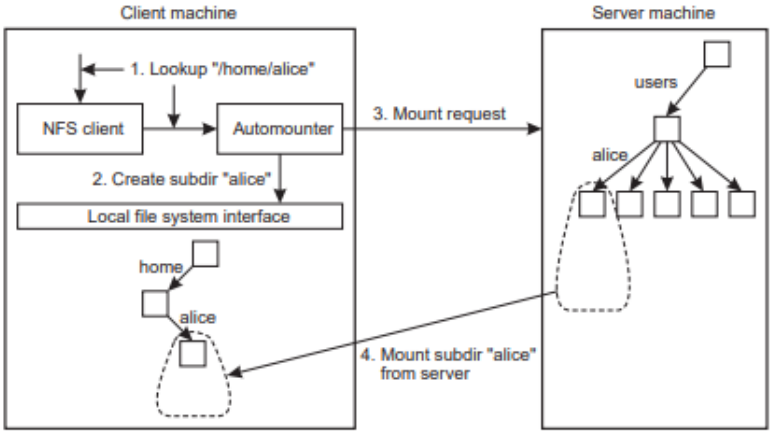
3.1.4.2 Traducción de nombres de ruta.

En NFS, los nombres de ruta no se pueden traducir en un servidor, porque el nombre puede cruzar un "punto de montaje" en el cliente; los directorios que contienen diferentes partes de un nombre de múltiples partes pueden residir en sistemas de archivos en diferentes servidores. Por lo tanto, los nombres de ruta se analizan, y su traducción se realiza de manera iterativa por el cliente. Cada parte de un nombre que se refiere a un directorio montado de forma remota se traduce a un manejador de archivos utilizando una solicitud de búsqueda separada al servidor remoto [4].

3.1.4.3 Automontador.

El cliente NFS detecta los intentos de acceder a puntos de montaje vacíos y los dirige al Automontador. El Automontador tiene una tabla de puntos de montaje y múltiples servidores candidatos para cada uno [4].

- Envía un mensaje de sondeo a cada servidor candidato y luego utiliza el servicio de montaje para montar el sistema de archivos en el primer servidor que responda.
- Mantiene la tabla de montajes pequeña y proporciona una forma simple de replicación para sistemas de archivos de solo lectura.



(*Distributed Systems Distributed File System [2] Distributed File System*, n.d.).

3.1.4.4 Seguridad del NFS con Kerberos.

La seguridad de las implementaciones de NFS se ha fortalecido mediante el uso del esquema Kerberos para autenticar clientes. En la implementación estándar original de NFS, la identidad del usuario se incluye en cada solicitud en forma de un identificador numérico sin cifrar. NFS no toma medidas adicionales para verificar la autenticidad del identificador proporcionado. Esto implica un alto grado de confianza en la integridad de la computadora cliente y su software por parte de NFS, mientras que el objetivo de Kerberos y otros sistemas de seguridad basados en autenticación es reducir al mínimo el rango de componentes en los que se asume confianza [4].

3.1.5 Operaciones del servicio de archivos NFS.

Las principales operaciones incluyen el montaje de sistemas de archivos remotos, la creación, lectura, escritura y eliminación de archivos o directorios, y la gestión de atributos con Getattr y Setattr. Cada solicitud de archivo se autentica para garantizar la seguridad, ya que NFS es un sistema sin estado. Además, se utiliza caching tanto en el cliente como en el servidor para mejorar el rendimiento, con mecanismos que aseguran la coherencia de los datos entre el servidor y los clientes. Como se muestra en la siguiente imagen:

lookup(dirfh, name) → *fh, attr*	Returns file handle and attributes for the file *name* in the directory *dirfh*.
create(dirfh, name, attr) → *newfh, attr*	Creates a new file *name* in directory *dirfh* with attributes *attr* and returns the new file handle and attributes.
remove(dirfh, name) → *status*	Removes file *name* from directory *dirfh*.
getattr(fh) → *attr*	Returns file attributes of file *fh*. (Similar to the UNIX *stat* system call.)
setattr(fh, attr) → *attr*	Sets the attributes (mode, user ID, group ID, size, access time and modify time of a file). Setting the size to 0 truncates the file.
read(fh, offset, count) → *attr, data*	Returns up to *count* bytes of data from a file starting at *offset*. Also returns the latest attributes of the file.
write(fh, offset, count, data) → *attr*	Writes *count* bytes of data to a file starting at *offset*. Returns the attributes of the file after the write has taken place.
rename(dirfh, name, todirfh, toname) → *status*	Changes the name of file *name* in directory *dirfh* to *toname* in directory *todirfh*.

(*Jisy Raju Assistant Professor, Distributed File Systems: Introduction*, 2004.)

3.2 CephFS (Ceph File System).

3.2.1 Descripción.

El sistema de archivos Ceph (CephFS) es un sistema de archivos compatible con los estándares POSIX que se construye sobre el almacenamiento de objetos distribuido de Ceph, llamado RADOS (Almacenamiento de Objetos Distribuidos Autónomos y Confiables). CephFS proporciona acceso a archivos a un clúster de almacenamiento Red Hat Ceph y utiliza la semántica POSIX siempre que sea posible. Por ejemplo, a diferencia de muchos otros sistemas de archivos en red comunes como NFS, CephFS mantiene una fuerte coherencia de caché entre los clientes. El objetivo es que los procesos que utilizan el sistema de archivos se comporten de la misma manera cuando están en diferentes hosts que cuando están en el mismo host. Sin embargo, en algunos casos, CephFS se desvía de las estrictas semánticas POSIX [8].

3.2.2 Funcionalidad.

Ceph funciona sobre RADOS (Reliable Autonomic Distributed Object Storage), donde los archivos se dividen en objetos que se distribuyen entre múltiples OSDs (Object Storage Daemons) mediante un algoritmo de hashing. Cada objeto es replicado en varios OSDs para asegurar su durabilidad. Cuando un cliente solicita un archivo, RADOS localiza y recupera los objetos necesarios, permitiendo un acceso eficiente incluso si algunos OSDs fallan.

El Ceph File System (CephFS) interactúa con RADOS a través de los MDS (Metadata Servers), que gestionan los metadatos y mantienen la coherencia de la caché entre los clientes. Al realizar operaciones de archivo, el MDS proporciona información actualizada sobre los objetos en RADOS, mientras que la caché en los OSDs y los clientes optimiza el acceso a los datos, garantizando que la información utilizada sea la más reciente.

3.2.3 Arquitectura

Como ya habíamos visto con anterioridad, los componentes de los sistemas de archivos distribuido tienen como base un servicio de archivos planos, un servicio de directorios y un módulo cliente. En el CephFS encontramos nuevos componentes que son:

3.2.3.1 RADOS (Reliable Autonomic Distributed Object Storage).

RADOS maneja la distribución de objetos en el clúster y proporciona mecanismos para la replicación, recuperación y gestión del almacenamiento [8].

3.2.3.2 Ceph OSD Daemons (Object Storage Daemons).

Cada OSD es responsable de almacenar objetos y de realizar operaciones de recuperación y replicación. Los OSDs gestionan los discos donde se almacenan los datos y son fundamentales para el rendimiento y la disponibilidad del sistema [8].

3.2.3.3 Ceph Monitors (MON).

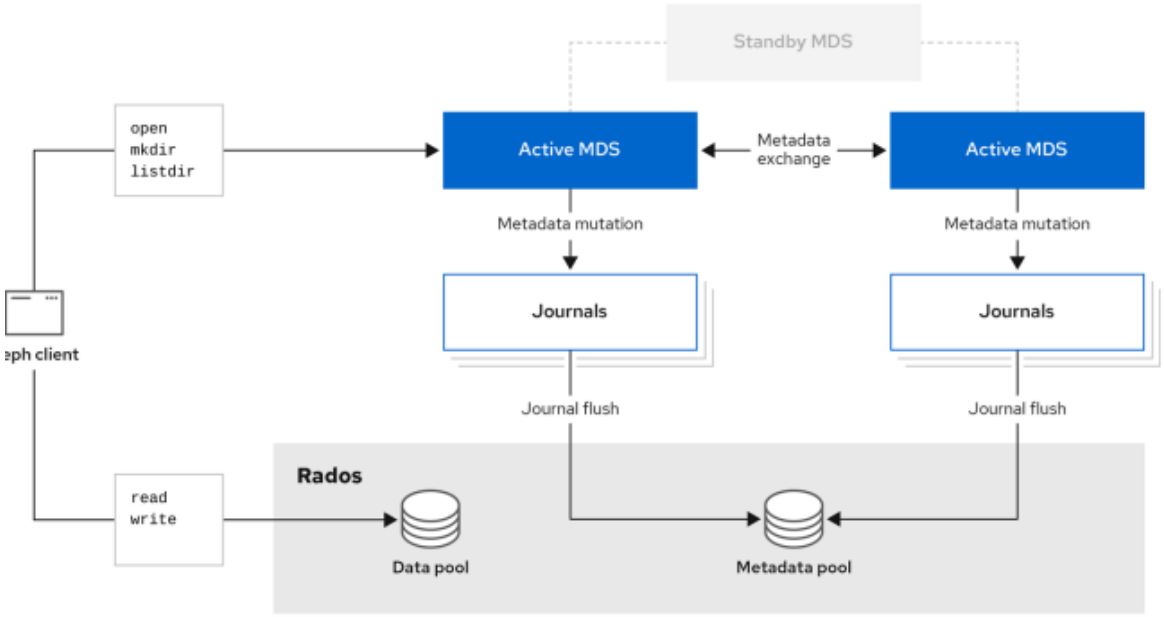
Los Monitores mantienen un mapa del clúster, supervisan la salud de los OSD y gestionan la información sobre la configuración del clúster. Se requieren al menos tres Monitores para asegurar la alta disponibilidad y la tolerancia a fallos [8].

3.2.3.4 Ceph Manager (MGR).

Proporciona funcionalidades adicionales como la gestión del clúster, estadísticas y el manejo de la interfaz de usuario. También puede incluir módulos que extienden las funcionalidades de Ceph [8].

3.2.3.5 Ceph Metadata Servers (MDS).

Los MDS gestionan las operaciones relacionadas con los metadatos de los archivos, como la creación, eliminación y renombrado de archivos y directorios [8].



(*Red Hat Ceph Storage 5 File System Guide Configuring and Mounting Ceph File Systems*, n.d.).

3.2.4 Características adicionales de CephFS.

3.2.4.1 Escalabilidad Horizontal.

CephFS permite agregar nodos de almacenamiento de manera sencilla, lo que facilita la expansión del clúster según las necesidades. La arquitectura está diseñada para escalar sin límites, lo que significa que se pueden añadir OSDs, Monitores y MDS adicionales sin afectar el rendimiento [8].

3.2.4.2 Alta Disponibilidad.

La replicación de datos en múltiples OSDs no solo proporciona durabilidad, sino que también asegura alta disponibilidad. CephFS puede seguir operando incluso si varios OSDs o incluso un Monitor fallan, gracias a su diseño redundante [8].

3.2.4.3 Soporte para Varios Protocolos.

CephFS es compatible con múltiples protocolos de acceso a datos, incluidos POSIX para sistemas de archivos, RBD (RADOS Block Device) para almacenamiento de bloques y librerías de objetos como S3 y Swift, lo que lo hace versátil para diversas aplicaciones [8].

3.2.4.4 Autorreparación.

CephFS puede detectar automáticamente fallos en los OSDs y otras componentes del clúster. Cuando se identifica un problema, el sistema inicia procesos de recuperación, como la re-replicación de objetos perdidos, para mantener la integridad de los datos [8].

3.2.4.5 Multitenencia.

CephFS permite a múltiples usuarios o aplicaciones acceder al mismo clúster de almacenamiento de manera segura y aislada, lo que lo hace adecuado para entornos de nube y soluciones de almacenamiento compartido [8].

4.- Aplicaciones de los sistemas de archivos distribuidos.

Los sistemas de archivos distribuidos tienen una amplia gama de aplicaciones en diferentes áreas tecnológicas, ya que permiten acceder a archivos y datos de forma remota, facilitando la colaboración, el almacenamiento masivo y el procesamiento distribuido. Algunas de las aplicaciones clave de los sistemas de archivos distribuidos incluyen:

4.1 Almacenamiento en la nube.

Servicios como Amazon S3, Google Drive o Dropbox utilizan sistemas de archivos distribuidos para proporcionar almacenamiento accesible desde cualquier lugar. Estos sistemas permiten a los usuarios subir, compartir y sincronizar archivos entre múltiples dispositivos.

En estos entornos, la capacidad de escalar fácilmente y la redundancia de datos son esenciales para ofrecer alta disponibilidad y durabilidad.

4.2 Big Data y análisis distribuido.

Herramientas como Hadoop Distributed File System (HDFS) se utilizan en aplicaciones de Big Data para gestionar grandes volúmenes de datos distribuidos entre múltiples nodos. Estos sistemas permiten el almacenamiento de grandes datasets y la ejecución de análisis paralelos, mejorando la eficiencia de procesamiento.

Se usan en aplicaciones como análisis de redes sociales, procesamiento de grandes logs de servidores y análisis de datos de sensores de IoT (Internet de las Cosas).

4.3 Entornos de virtualización y contenedores.

Las plataformas de virtualización como VMware, OpenStack, y sistemas de orquestación de contenedores como Kubernetes, dependen de sistemas de archivos distribuidos para proporcionar almacenamiento persistente a máquinas virtuales y contenedores.

5.- Conclusiones.

Los sistemas de archivos distribuidos, como NFS y CephFS, son opciones potentes que se adaptan a distintas necesidades, y su elección depende mucho del tipo de proyecto y de los objetivos que tengamos en mente. Al considerar soluciones como estas, lo que más me llamó la atención es su capacidad para manejar grandes volúmenes de datos y facilitar el acceso a múltiples usuarios sin complicar demasiado la gestión. Estas tecnologías son clave para cualquier infraestructura que busque optimizar el rendimiento y la disponibilidad de datos de manera eficiente.

Lo que me resultó más interesante de NFS es su simplicidad y facilidad de uso. Si bien tiene limitaciones en términos de escalabilidad, para proyectos más pequeños o con menos demandas es una opción rápida y confiable que permite compartir archivos sin demasiadas complicaciones. CephFS es fascinante por su escalabilidad masiva y su enfoque completamente distribuido, lo que lo hace perfecto para proyectos ambiciosos que buscan asegurar la tolerancia a fallos y la recuperación automática.

Lo que más me atrajo de ambos sistemas es cómo, dependiendo del contexto, pueden ser herramientas muy útiles. NFS me parece ideal cuando lo que se busca es simplicidad y funcionalidad inmediata, mientras que CephFS es impresionante en términos de robustez y flexibilidad para proyectos más complejos. Cada sistema tiene su lugar según el tamaño y las necesidades del entorno, lo que los convierte en piezas fundamentales en la planificación de cualquier infraestructura tecnológica.

"The network is the computer." (Sun Microsystems, 1999).

6.- Referencias Bibliográficas.

1. *Jisy Raju Assistant Professor, CE Cherthala Module 4 Distributed file system: File service architecture -Network file system-Andrew file system- Name Service 4.1 Distributed file Systems: Introduction*. (n.d.). http://www.cectl.ac.in/images/pdf_docs/studymaterial/cse/s7/DC4.pdf

2. ¿Qué es DFS? - Sistemas de archivos distribuidos. (2023). https://www.nutanix.com/es/info/distributed-file-systems
3. micronimics. (2022, December 24). Advantages and Disadvantages of HDFS And Traditional File Systems - Data Recovery in Ahmedabad, Best Data Recovery in India. Data Recovery in Ahmedabad, Best Data Recovery in India. https://www.micronicsindia.com/advantages-and-disadvantages-of-hdfs-and-traditional-file-systems/
4. Jisy Raju Assistant Professor, CE Cherthala Module 4 Distributed file system: File service architecture -Network file system-Andrew file system- Name Service 4.1 Distributed file Systems: Introduction. (n.d.). http://www.cectl.ac.in/images/pdf_docs/studymaterial/cse/s7/DC4.pdf
5. Kozlowski, S. W., & Bell, B. S. (2007). A theory-based approach for designing distributed learning systems.
6. *Distributed Systems Distributed File System [2] Distributed File System*. (n.d.). https://www.ia.pw.edu.pl/~tkruk/edu/rso.b/lecture/pre/rso08_pre.pdf
7. Murugesan, P. (n.d.). *Distributed File Systems*. Retrieved October 12, 2024, from https://www.engr.colostate.edu/ECE658/2013/onlinepresentation/Prabhakaran/Prabhakaran.pdf
8. *Red Hat Ceph Storage 5 File System Guide Configuring and Mounting Ceph File Systems*. (n.d.). Retrieved October 12, 2024, from https://docs.redhat.com/ja/documentation/red_hat_ceph_storage/5/pdf/file_system_guide/red_hat_ceph_storage-5-file_system_guide-ja-jp.pdf