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 [1].

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.

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].

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 [3].

Desventajas:

Más lento que los sistemas de archivos locales: DFS no es tan rápido como algunos otros sistemas de archivos, como los sistemas de archivos locales. 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].

Actualizaciones o cambios de archivos: las actualizaciones o cambios a archivos, no se reflejan en tiempo real, debido a que no se pueden gestionar documentos que se encuentren abiertos por otros usuarios.

Costos: La implementación y mantenimiento de un DFS requiere inversión en infraestructura y recursos, como hardware y software [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, el servicio de archivos planos no tiene operaciones de apertura y cierre; los archivos pueden ser accedidos inmediatamente citando el UFID correspondiente. La interfaz del 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.

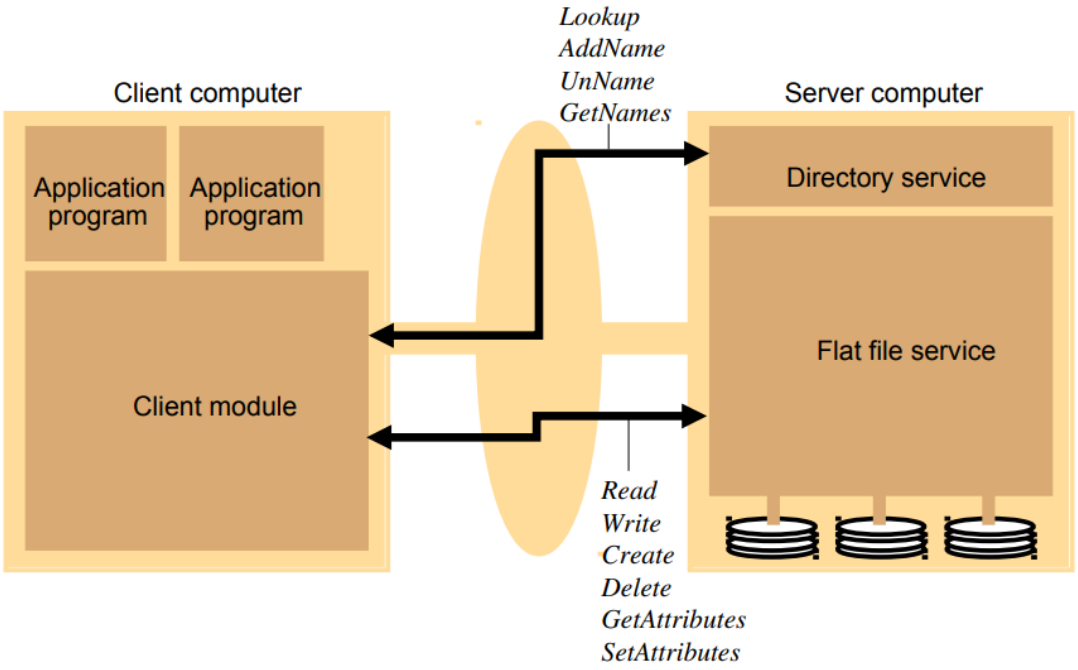


Imagen 3 *(Jisy Raju Assistant Professor, Distributed File Systems: Introduction, 2004).*

3.- Ejemplo de Sistemas de archivos distribuidos.

3.1 CephFS (Ceph File System).

3.1.1 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. 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].

3.1.2 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.

3.1.3 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:

3.1.3.1 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].

3.1.3.2 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].

3.1.3.3 Ceph Monitors (MON).

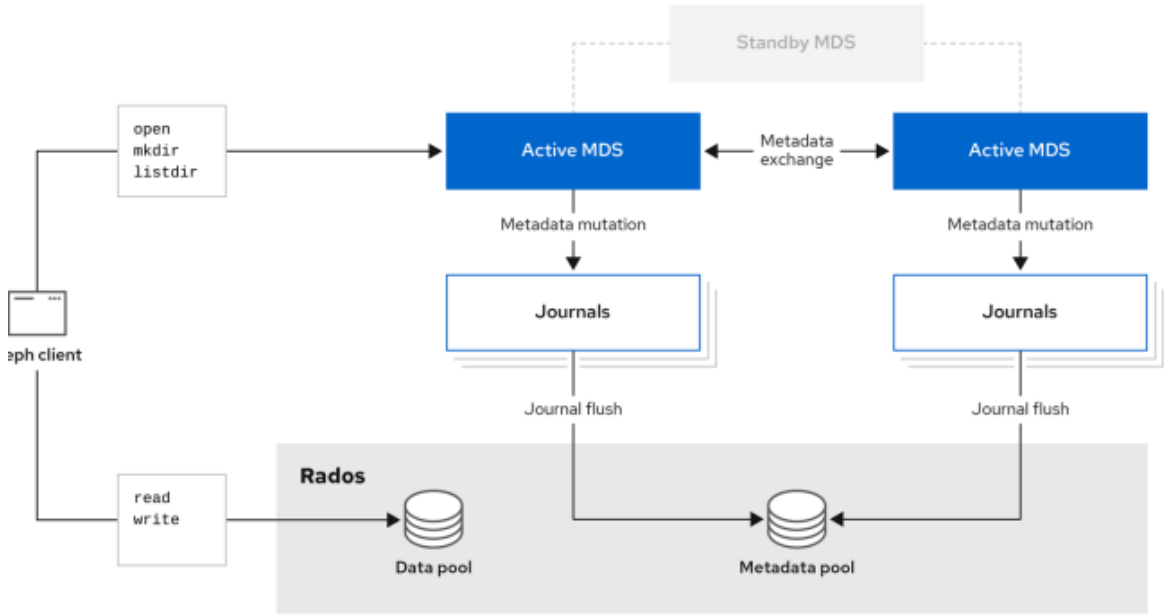
Los Monitores mantienen un mapa del clúster, supervisan la salud de los OSD y gestionan la información sobre la configuración del clúster. Se requieren al menos tres Monitores para asegurar la alta disponibilidad y la tolerancia a fallos [8].

3.1.3.4 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].

3.1.3.5 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.).

3.1.4 Características adicionales de CephFS.

3.1.4.1 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].

3.1.4.2 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].

3.1.4.3 Soporte para Varios Protocolos.

CephFS es compatible con múltiples protocolos de acceso a datos, 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].

3.1.4.4 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].

3.1.4.5 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 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.

4.2 Almacenamiento en la nube y Big data.

Aquí se muestran las principales fortalezas y debilidades que tienen los sistemas de archivos distribuidos en aplicaciones como el almacenamiento de la nube y el big data:

DFS	Fortalezas	Debilidades
S3 (Amazon)	Su fortaleza está en su escalabilidad y alta disponibilidad, con un enfoque claro en grandes volúmenes de datos para aplicaciones empresariales.	Puede ser complejo de usar y más costoso a largo plazo.
Google Drive	Muy fácil de usar y se integra bien con otros servicios de Google.	Limitaciones cuando se trata de manejar grandes volúmenes de datos y su escalabilidad está más enfocada en uso personal o empresarial pequeño.
Dropbox	Facilidad de uso y aplicabilidad.	Limitado para grandes necesidades empresariales o big data.
Hadoop (HDFS)	Excelencia en procesamiento y almacenamiento distribuido de big data e ideal para manejar enormes volúmenes de datos.	Requiere experiencia técnica para implementar y gestionar.
CephFS	Ofrece flexibilidad y rendimiento ajustable para infraestructuras empresariales complejas, con soporte tanto para almacenamiento de bloques como objetos.	Su configuración puede ser técnica y demandante.

5.- Conclusiones.

Los sistemas de archivos distribuidos, como 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.

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.

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