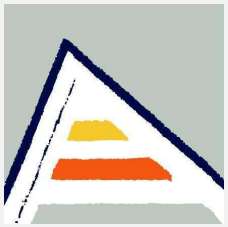


Grado en Ingeniería Informática

Sistemas distribuidos

Sistema de archivos distribuido

Víctor Vives
vvives@dtic.ua.es

Departamento de Tecnología Informática y Computación

2021 - 2022

1. Introducción

¿Qué es un sistema de archivos distribuido?

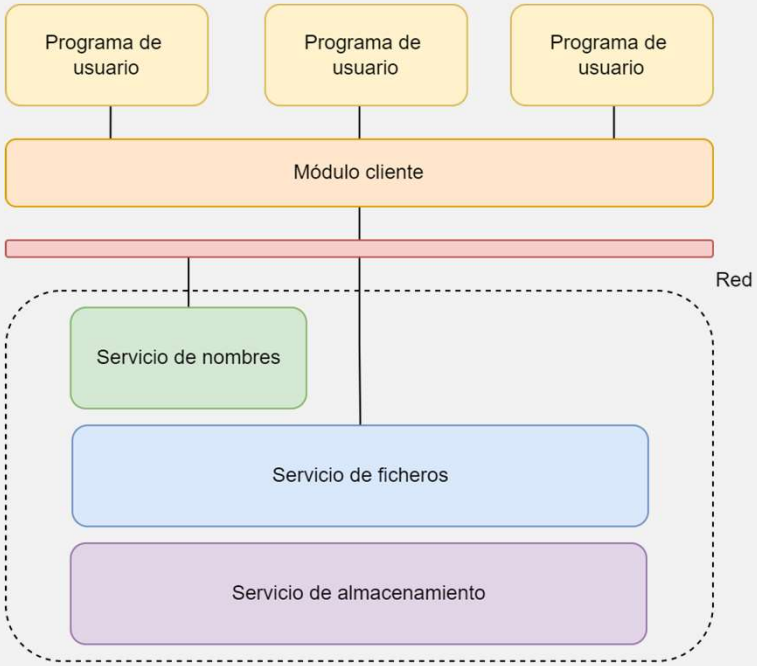
Es un sistema de archivos con datos almacenado en un servidor.

- También DFS o *Distributed File System*.
- Se accede a los datos y se procesan como si estuviesen almacenados en local.
- Permite compartir archivos entre usuarios de una forma **controlada** y **autorizada**.
    - Sin notar pérdidas de rendimiento.
    - Acceso de forma simultánea e ininterrumpida.

2. Diseño

Componentes de un sistema de archivos distribuido

- Servicio de almacenamiento
- Servicio de ficheros
- Servicio de nombres o de directorio
- Módulo cliente: biblioteca de clases o API



2. Diseño

Objetivos de diseño

- Transparencia
- Movilidad de los usuarios
- Rendimiento
- Alta disponibilidad y tolerancia a fallos
- Concurrencia
- Fiabilidad e integridad de la información
- Seguridad
- Heterogeneidad del hardware y el software

2. Diseño

Objetivos de diseño

Transparencia

- Transparencia de **estructura**: no es necesario que el cliente conozca el número o la ubicación de los servidores de archivos y los dispositivos de almacenamiento. Se deben proporcionar varios servidores de archivos para el rendimiento, la adaptabilidad y la confiabilidad.

- Transparencia de **acceso**: tanto los archivos locales como los remotos deben ser accesibles de la misma manera. El sistema de archivos debe ubicarse automáticamente en el archivo al que se accede y enviarlo al lado del cliente.

2. Diseño

Objetivos de diseño

Transparencia

- Transparencia de **nombres**: no debe haber ninguna pista en el nombre de archivo a la ubicación del archivo. Una vez que se le da un nombre al archivo no debe cambiarse durante la transferencia de un nodo a otro.

- Transparencia de **replicación**: Si un archivo se copia en varios nodos, tanto las copias del archivo como sus ubicaciones deben ocultarse de un nodo a otro.

2. Diseño

Objetivos de diseño

Movilidad de los usuarios

Llevará automáticamente el directorio de inicio del usuario al nodo donde el usuario inicia sesión.

Rendimiento

El rendimiento se basa en la cantidad de tiempo promedio necesario para convencer a las solicitudes del cliente. Este tiempo cubre el tiempo de la CPU, más el tiempo necesario para acceder al almacenamiento secundario, más el tiempo de acceso a la red. Es aconsejable que el rendimiento del sistema de archivos distribuido sea similar al de un sistema de archivos centralizado.

2. Diseño

Objetivos de diseño

Alta disponibilidad

Un sistema de archivos distribuido debería poder continuar en caso de fallos parciales. Un sistema de archivos altamente auténtico y adaptable debe tener servidores de archivos diferentes e independientes para controlar dispositivos de almacenamiento diferentes e independientes.

Concurrencia y escalabilidad

El sistema de archivos distribuido de escalar rápidamente a medida que crece la cantidad de nodos y usuarios del sistema. El servicio no debe interrumpirse sustancialmente ante estos aumentos.

2. Diseño

Objetivos de diseño

Fiabilidad

La probabilidad de pérdida de datos debe minimizarse tanto como sea posible. Un sistema de archivos distribuido debe crear copias de seguridad de los archivos clave si se pierde el original.

Integridad de los datos

Las solicitudes de acceso concurrente por parte de varios usuarios que compiten por el acceso al mismo archivo deben sincronizarse correctamente utilizando métodos de control de la concurrencia.

2. Diseño

Objetivos de diseño

Seguridad

Para proteger la información del sistema de archivos distribuido de accesos no deseados y no autorizados se deben implementar mecanismos de seguridad para ello.

Heterogeneidad

Los usuarios deben tener la posibilidad de utilizar múltiples plataformas.

3. Modelos

Modelos de acceso a ficheros remotos

- Modelo de servicio remoto
  - Las operaciones se realizan en los servidores
  - Problemas de eficiencia
- Modelo de caché de datos
  - Se accede a los ficheros de forma local
  - Aumento del rendimiento
  - Problemas de consistencia
- Combinación de ambos modelos

3. Modelos

Modelo de servicio remoto

Servidores **sin estado** o *stateless*

No almacenan información entre solicitudes de un mismo cliente

- Tolerancia a fallos
- No requiere de llamadas para la apertura y cierre de ficheros
- No se desperdicia la memoria en tablas
- No existe límite para el número de ficheros en uso
- No se producen problemas si cae un cliente

3. Modelos

Modelo de servicio remoto

Servidores **con estado** o *stateful*

Almacenan información entre solicitudes de un mismo cliente

- Mensajes de solicitud de servicio más cortos
- Mejor rendimiento
- Es posible realizar operaciones de lectura anticipada
- Permiten el bloqueo de ficheros

3. Modelos

Modelo de caché de datos

El objetivo es retener en memoria principal aquellos datos que han sido usados recientemente.

Localización de la caché

Hay cuatro lugares posibles en los que almacenar la información.

1. En el disco del servidor
2. En la memoria del servidor
3. En el disco del cliente
4. En la memoria del cliente

3. Modelos

Localización de la caché

1. En el disco del servidor

No se aconseja la caché en disco del servidor, demasiado costoso.

2. En la memoria del servidor

- Es transparente para los clientes
- Los SO tradicionales suelen utilizar este esquema (NFS)
- Coste de las transferencias por red

3. Modelos

Localización de la caché

3. En el disco del cliente

- Fiabilidad y gran tamaño de la caché.
- Hay que acceder al disco

4. En la memoria del cliente

- Se eliminan los costes de transmisión por la red y de acceso a disco
- Máximo rendimiento en caso de acierto
- Problemas de inconsistencia de la información en la caché

3. Modelos

Localización de la caché

4. En la memoria del cliente

Problemas de inconsistencia de la información en la caché

- Se debe considerar cuándo propagar las modificaciones hechas en la caché al fichero en el servidor.
- Se debe considerar cómo verificar la validez de los datos en las cachés.

Existen cuatro esquemas básicos para propagar las modificaciones.

- Escritura simultánea (*write-through*)
- Escritura retrasada (*delayed-write*)
- Escritura al cerrar (*write-on-close*)
- Control centralizado

3. Modelos

Localización de la caché

4. En la memoria del cliente

Escritura simultánea (*write-through*)

- Cuando un usuario edita una entrada de caché se escribe inmediatamente en el servidor.
- Cualquier procedimiento que requiera un archivo del servidor siempre recibirá la información más actualizada.
- En caso de inconsistencia de comparan los tiempos de modificación de las copias.

Escritura retrasada (*delayed-write*)

- Para reducir el tráfico continuo de la red realiza actualizaciones periódicas o por lotes en el servidor.
- Mejora el rendimiento al permitir una única operación de escritura masiva en lugar de varias pequeñas.

3. Modelos

Localización de la caché

4. En la memoria del cliente

Escritura al cerrar (*write-on-close*)

- Un paso más avanzado es escribir el archivo en el servidor una vez se haya cerrado.
- La segunda escritura sobrescribe la primera si dos archivos en caché se escriben uno al lado del otro.

Control centralizado

- Con fines de seguimiento el cliente envía información sobre los archivos que acaba de abrir al servidor.
- Si dos o más archivos tratan de abrir un archivo previamente abierto para escritura se deniega el acceso.
- Una vez se ha finalizado la escritura en servidor y actualiza la tabla correspondiente se permite el acceso.

3. Modelos

Localización de la caché

Validación de modificaciones

La política de modificaciones especifica cuándo se actualiza la copia principal de un fichero cuando una de las copias en caché es modificada, pero no establece cuándo modificar las cachés.

El contenido de una caché se vuelve inválido cuando otro cliente modifica la copia principal:

- Es necesario comprobar si la caché de un cliente es consistente respecto a la copia principal.
- En caso contrario es necesario invalidar la caché y actualizar los datos.

Existen dos estrategias básicas: iniciada por el **cliente** e iniciada por el **servidor**.

3. Modelos

Localización de la caché

Validaciones iniciadas por el servidor

Cuando un cliente quiere abrir un archivo informa al servidor de archivos de la intención de la apertura: lectura, escritura o ambos. El servidor de archivos realiza entonces un **seguimiento** de qué cliente está trabajando en el archivo y en qué modo. Cuando detecta inconsistencias entonces reacciona:

- Un cliente notifica al servidor del cierre de un archivo así como los cambios realizados antes del cierre. Entonces el servidor actualiza su base de datos para reflejar qué clientes tienen el archivo abierto y de qué modo.
- El servidor puede **denegar** o **encolar** una solicitud o deshabilitar el almacenamiento en caché solicitando que todos los clientes que tienen el archivo abierto lo eliminen de sus cachés locales para detectar cualquier inconsistencia en cada uno de los clientes.

4. Réplicas

Replicación de ficheros

Un **fichero replicado** es aquel del que existen **varias copias** cada una de las cuáles de un **servidor diferente**.

La replicación de ficheros aporta las siguientes ventajas:

- Aumenta la disponibilidad
- Aumenta la fiabilidad
- Mejora el tiempo de respuesta
- Reduce el tráfico de la red
- Mejora el rendimiento
- Beneficia la escalabilidad
- Permite trabajar en modo operación desconectada

4. Réplicas

Replicación de ficheros

Es necesario considerar:

- Cómo conseguir que la replicación sea transparente a los usuarios.
- Cómo actualizar las copias en el caso de modificaciones en una réplica.
- Cómo denominar a las réplicas.
- Cómo controlar la replicación.

¿Cómo distinguir una réplica de otra si ambas tienen el mismo identificador?

Un servidor de nombres debería hacer corresponder al identificador la réplica más conveniente.

4. Réplicas

Replicación de ficheros

Protocolos de actualización de réplicas.

- Solo lectura (read-only): se aplica a ficheros inmutables.
- Escribir en todos - leer de cualquiera: escrituras costosas y problemas cuando un servidor se cae.
- Disponibilidad de copias: problemas de inconsistencias en particiones de red.
- Copia primaria (write-once-read-many): problemas si cae el servidor primario.
- Basado en quorum: asegura que dos copias no se leen o escriben por dos transacciones de forma concurrente.

5. Aplicaciones

NFS o *Network File System*

NFS es una arquitectura cliente-servidor que permite al usuario de una máquina ver, almacenar y actualizar archivos de forma remota. El protocolo NFS es uno de los varios estándares de sistemas de archivos distribuidos para el almacenamiento conectado en red (NAS).

SMB o *Server Message Block*

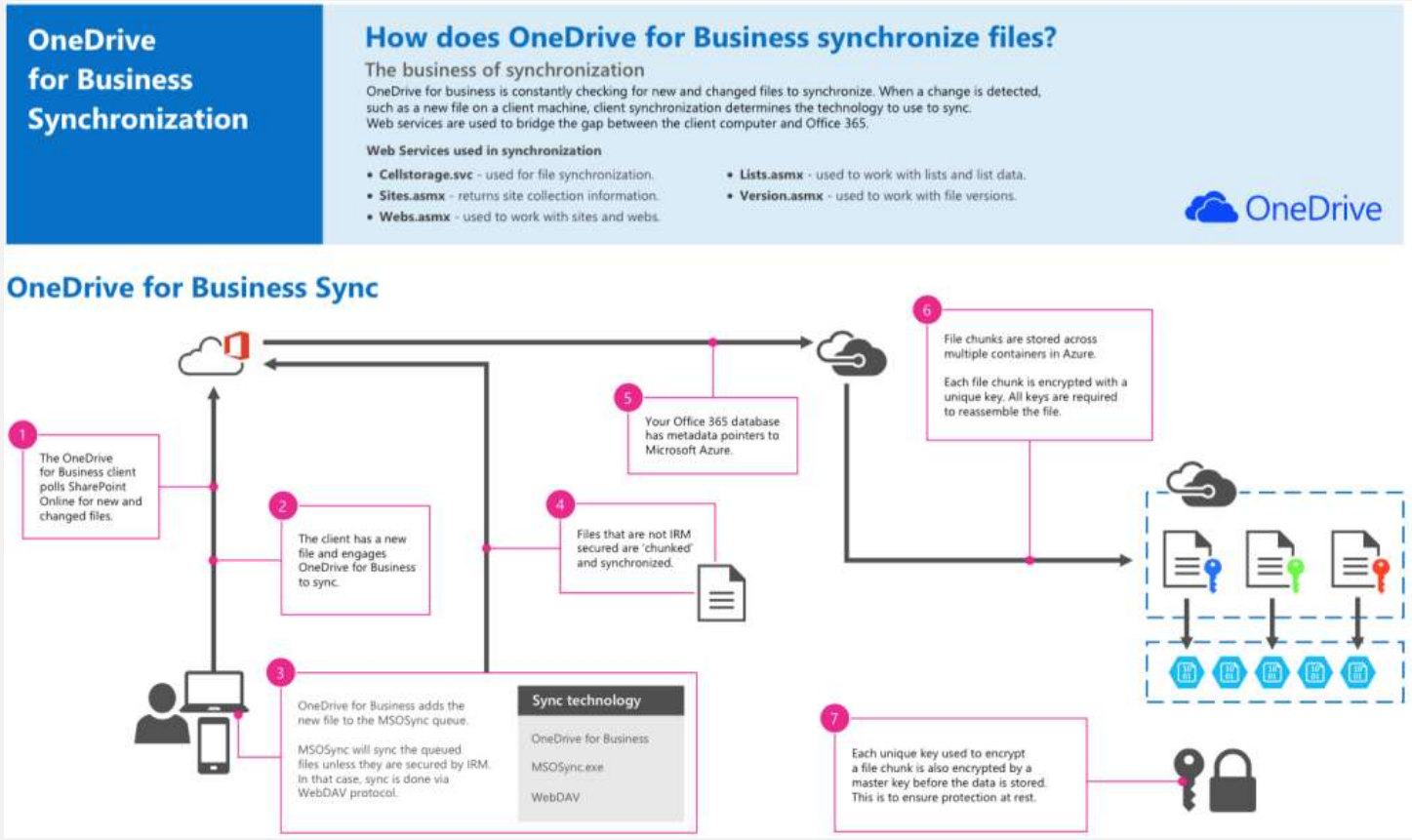
SMB, o bloque de mensajes del servidor, es un protocolo para compartir un archivo inventado por IBM. El protocolo SMB se creó para permitir que una máquina pueda realizar operaciones de lectura y escritura en archivos de un host remoto a través de una red de área local (LAN). Se puede acceder a los directorios en el host remoto a través de SMB y se denominan recursos compartidos.

5. Aplicaciones

Diferencias entre NFS y SMB

- SMB no distingue entre mayúsculas y minúsculas (*case sensitive*) mientras que NFS sí.
- NFS es generalmente más rápido cuando se lee o escriben archivos pequeños.
- NFS también es más rápido en la navegación entre directorios.
- NFS utiliza un sistema de autenticación basado en host y SMB basado en usuario.
- NFS es más sencillo de configurar que SMB.
- NFS es adecuado para usuarios de Linux mientras que SMB lo es para usuarios de Windows.
- En redes domésticas se recomienda NFS sin cifrar en Linux para máximo rendimiento.

5. Aplicaciones



5. Aplicaciones

HDFS o *Hadoop Distributed File System*

Hadoop es un grupo de servicios de software de código abierto utilizado para soluciones **Big Data**. Proporciona un framework para el **almacenamiento distribuido** utilizando el modelo de programación *MapReduce*.

El núcleo de Hadoop contiene una parte de **almacenamiento** conocida como HDFS y una parte **operativa** que es el MapReduce.

5. Aplicaciones

HDFS o *Hadoop Distributed File System*

Componentes

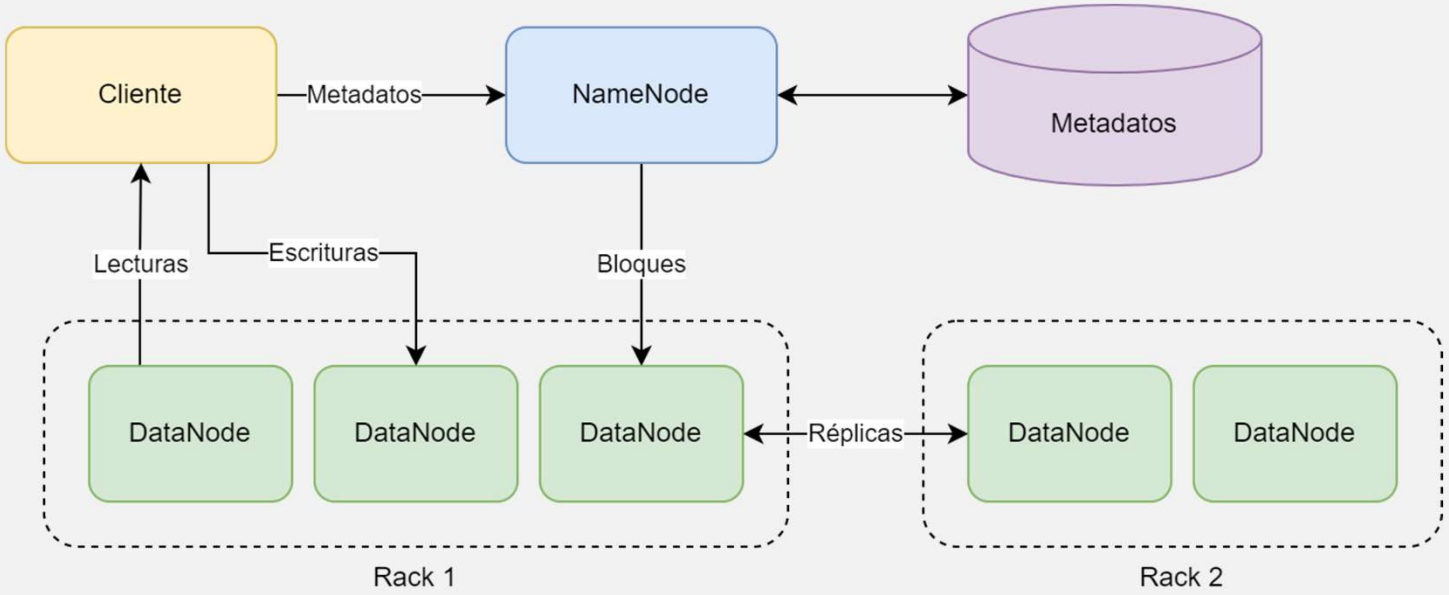
El *NameNode* (NN) es el **nodo principal** o nodo maestro del sistema. No se encarga de almacenar los datos sino de gestionar su **acceso** y de almacenar los **metadatos**. Necesita menos espacio en disco pero más recursos en términos de memoria y cómputo que los DataNodes.

Los *DataNodes* (DN) son los **nodos del clúster que almacenan los datos**. Se encarga de gestionar el almacenamiento del nodo. Se suele utilizar hardware básico con varios discos y gran capacidad. Su tipología permite **escalar el sistema de forma horizontal de forma efectiva y con bajo coste**.

HDFS utiliza un sistema de lectura y escritura denominado *Write once read many*.

5. Aplicaciones

HDFS o *Hadoop Distributed File System*



5. Aplicaciones

HDFS o *Hadoop Distributed File System*

Alta escalabilidad

HDFS utiliza almacenamiento local que escala horizontalmente soportando miles de nodos.

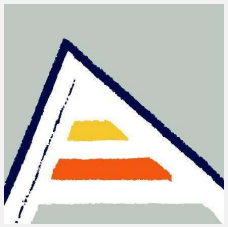
Lo típico es desplegar decenas o cientos de nodos para manejar cientos de terabytes.

Capacidad para escalar a decenas de petabytes.

Integridad de los datos

HDFS almacena por defecto tres copias de cada bloque de datos.

Aumenta el espacio y el coste de infraestructura pero evita la necesidad de replicación.



Grado en Ingeniería Informática

Sistemas distribuidos

Sistema de archivos distribuido

Víctor Vives
vvives@dtic.ua.es

Departamento de Tecnología Informática y Computación

2021 - 2022