
Tema 3:

Sistema de archivos distribuidos AFS (Andrew File System)

AFS: Introducción

- Sistema de ficheros distribuido que proporciona acceso transparente a archivos remotos para los programas UNIX que se ejecutan en una estación de trabajo
- Los servidores AFS mantienen los archivos UNIX “locales”, pero el sistema de ficheros en los servidores está basado en NFS.
- La principal ventaja de AFS frente a NFS es la “escalabilidad”
 - AFS está diseñado para trabajar bien con un gran número de usuarios
 - Utiliza una cache de archivos completos en los nodos cliente

- Principales características de AFS

- Servicio de transferencia de archivos completos

- Los servidores AFS transfieren a los clientes ficheros o directorios completos
 - AFS-3 puede transferir archivos de hasta 64Kbyte
 - Los mayores a 64Kbyte son transferidos a trozos

- Cache de archivos completos

- Los archivos se mantienen en una cache en el disco local
 - La cache contiene varios cientos de archivos
 - Los utilizados más recientemente en el cliente
 - La caché es permanente sobreviviendo a los arranques
 - Los `open` de ficheros utilizan preferentemente copias locales frente a las remotas.

■ Funcionamiento

- Cuando un proceso en el cliente solicita un fichero que no posee, hace una llamada al servidor para copiar el fichero
- El fichero se almacena en el sistema de archivos local
- Todas las operaciones `read`, `write`, etc, se aplican a la copia local
- Cuando se cierra el fichero, si se ha cambiado se actualiza en el servidor y la copia local permanece en el almacenamiento local

■ Consideraciones

- La abrumadora mayoría de accesos se hace sobre
 - ficheros que son actualizados infrecuentemente (códigos de comandos, librerías, etc)
 - No se actualizan
 - ficheros accedidos normalmente por un solo usuario (todos los del directorio `/home`)
 - Si se actualizan la copia es actualizada estará en el cliente del propietario
- La caché local puede reservar una proporción sustancial del espacio de disco en cada estación, (100Mb, 1Gb)
 - Esto será suficiente para almacenar todos los ficheros utilizados por el usuario
- Las observaciones de cargas de trabajo UNIX típicas en diferentes entornos se utilizarán para hacer suposiciones sobre
 - Tamaños promedio de ficheros
 - Máximo de archivos y
 - Proximidad de referencia

- Consideraciones

- Observaciones

- Los archivos son pequeños
 - La mayoría <10kbyte
 - Las operaciones de lectura son mucho más frecuentes que las de escritura
 - Unas 6 veces más habituales
 - El acceso secuencial es lo habitual, y el aleatorio es inusual
 - La mayoría de ficheros son leídos y escritos por sólo un usuario
 - Cuando se comparte un fichero normalmente solo un usuario lo modifica
 - Los archivos son referenciados a ráfagas.
 - Si un archivo ha sido referenciado recientemente hay una alta probabilidad de que sea referenciado en un futuro próximo

- Consideraciones

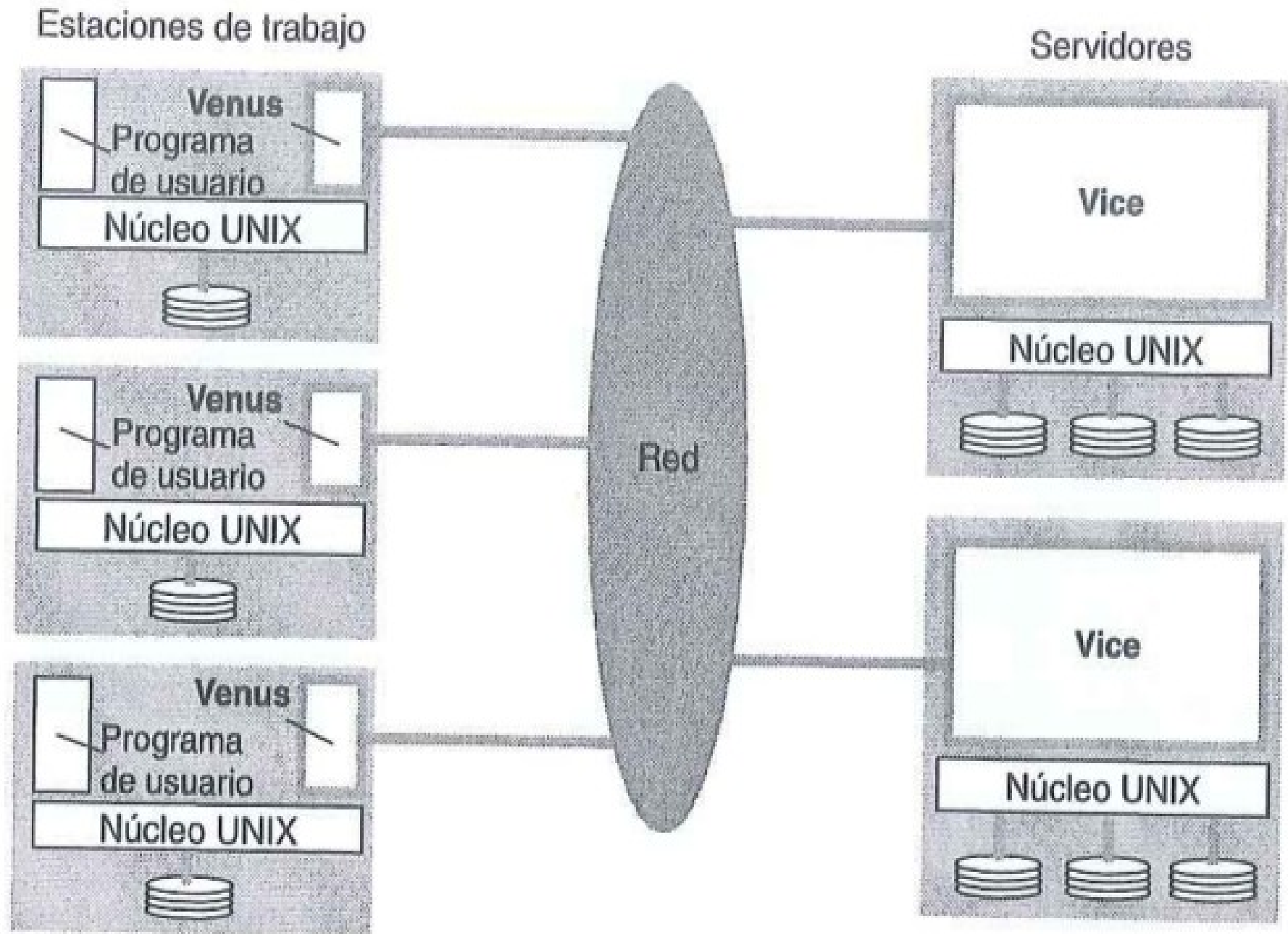
- Bases de datos

- Existe modelos de ficheros distribuidos donde no se dan las observaciones anteriores, como las bases de datos distribuidas.
 - Son compartidas por muchos usuarios
 - Son actualizadas muy a menudo
 - AFS no contempla en su diseño facilidades para trabajar con bases de datos
 - Debería contemplarse otra estrategia

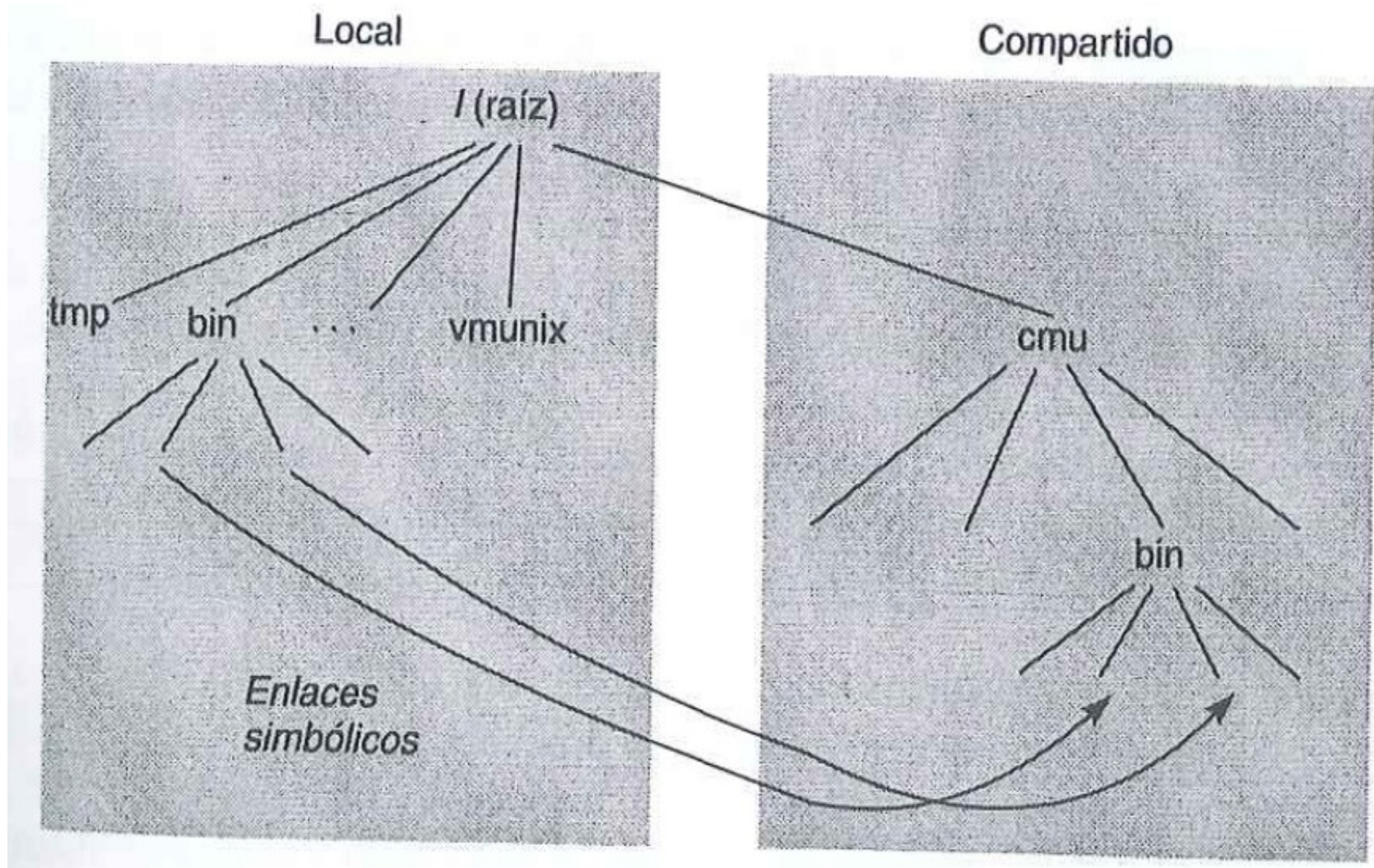
AFS: Implementación

- Implementado como dos procesos
 - Vice → Módulo servidor que se ejecuta a nivel de usuario en cada servidor
 - Venus → Módulo cliente que se ejecuta a nivel de usuario en cada cliente
- Los archivos locales se manejan como archivos normales UNIX
- Los archivos remotos se almacenan en los servidores y son transferidos a las caches de los discos locales de los clientes
- Sigue una jerarquía convencional de directorios UNIX con un subárbol específico `/cmu` que contiene todos los archivos compartidos
 - Los ficheros locales sólo se utilizan para ficheros
 - Temporales
 - Necesarios para el arranque
 - `/bin /lib` etc, están montados como enlaces simbólicos al espacio compartido
 - Los directorios de trabajo de los usuarios `/home` están en espacio compartido para que los usuarios accedan a ellos desde cualquier cliente

AFS: Implementación



AFS: Implementación



AFS: Implementación

- El núcleo UNIX es una versión modificada de UNIX BSD
 - Intercepta las operaciones `open`, `close`, etc cuando se refieren a ficheros en el espacio compartido y las pasan al proceso `Venus` del cliente

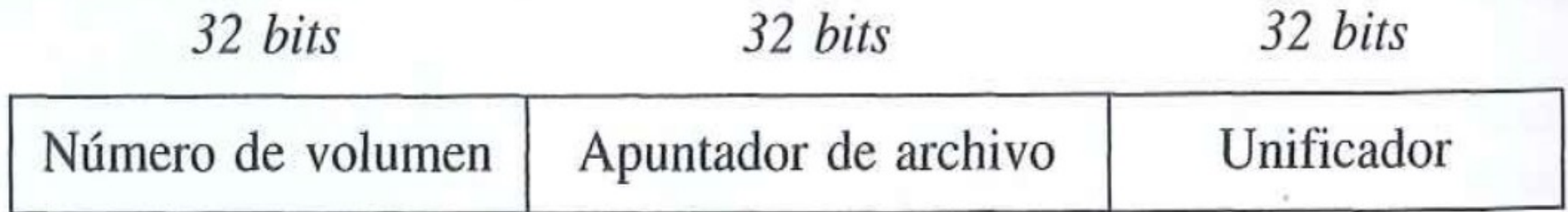


AFS: Implementación

- En el disco local se usa una partición del disco como una cache que mantiene las copias de los ficheros remotos
 - Venus administra dicha cache
 - Elimina archivos menos usados cuando se requiere espacio para traer nuevos ficheros.
 - Es suficientemente grande como para tener varios cientos de ficheros
 - Esto hace que los clientes trabajen de forma independiente al servidor
- Los archivos se agrupan en volúmenes para facilitar la localización y mantenimiento
 - Son más pequeños que los volúmenes UNIX que son la unidad de agrupamiento de NFS
 - Se mantienen en volúmenes separados
 - Los ficheros personales de cada usuario
 - Los ficheros binarios del sistema
 - Los ficheros de documentación
 - Los ficheros de la bibliotecas

AFS: Implementación


- Cada fichero o directorio se identifica con un identificador de archivo único, “**ida**”, de 96bits semejante al UFID.
 - Venus traduce los nombres de ruta emitidos por los clientes a **idas**
 - Cada ida está compuesto por



- Mientras los usuarios utilizan los nombres UNIX, AFS usa los idas en la comunicación `Venus → Vice`
 - Venus traduce los nombres UNIX a idas y los pasa a los Vice para su búsqueda

AFS: Implementación

- Llamadas al sistema en AFS

Proceso de usuario	Núcleo UNIX	Venus	Red	Vice
<i>open(NombreArchivo, modo)</i>	<p>Si <i>NombreArchivo</i> se refiere a un archivo del espacio de archivos compartido, pasa la petición a Venus.</p> <p>Abre el archivo local y devuelve el descriptor de archivo a la aplicación.</p>	<p>Comprueba la lista de archivos en la caché local. Si no está presente o no hay una <i>promesa de devolución de llamada</i>, envía una petición de archivo al servidor Vice que es el custodio del volumen que contiene el archivo.</p> <p>Sitúa la copia del archivo en el sistema de archivos local, inserta su nombre local en la caché de lista local y devuelve el nombre local a UNIX.</p>		<p>Transfiere una copia del archivo y una <i>promesa de devolución de llamada</i> a la estación de trabajo. Registra la <i>promesa de devolución de llamada</i>.</p>

AFS: Implementación

- Llamadas al sistema en AFS

<i>read(DescriptorArchivo, Búfer, tamaño)</i>	Realiza una operación de lectura UNIX normal sobre la copia local.			
<i>write(DescriptorArchivo, Búfer, tamaño)</i>	Realiza una operación de escritura UNIX normal sobre la copia local.			
<i>close(DescriptorArchivo)</i>	Cierra la copia local y notifica a Venus que el archivo ha sido cerrado.	Si la copia local ha sido cambiada, envía una copia al servidor Vice que es el custodio del archivo.	→	Reemplaza los contenidos del archivo y envía una <i>devolución de llamada</i> a todos los otros clientes que posean <i>promesas de devolución de llamada</i> sobre el archivo.

AFS: Consistencia de la cache

- Cuando Vice proporciona una copia de un archivo a Venus también le proporciona una *promesa de devolución de llamada*
 - Es un testigo que garantiza que Vice notificará a Venus cuando otro cliente modifica el archivo
 - Las *promesa de devolución de llamada* se almacenan en la caché junto con los archivos
 - Tienen dos estados: *válida y cancelada*
 - Cuando un servidor modifica un archivo el Vice hace una *devolución de llamada* a cada Venus con *promesa de devolución de llamada* y la pone a *cancelada*
 - Cuando el cliente hace un open, Venus comprueba el testigo.
 - Valido → utiliza la copia de la cache
 - Cancelado → solicita copia a Vice

AFS: Consistencia de la cache

- Cuando una estación de trabajo se rearranca pueden haberse perdido algunas *devoluciones de llamada*. Así que Vice para cada fichero de la cache con testigo válido Venus hace lo siguiente
 - Venus envía una petición de validación de la cache a Vice con el tiempo de modificación
 - Si la marca de tiempo coincide con la del servidor, se valida
 - Si la marca de tiempo no coincide con la del servidor, se cancela
- Para prevenir algunas posibles pérdidas de comunicación, las promesas de devolución de llamada deben ser renovadas antes de cada `open` si ha transcurrido un tiempo T , normalmente del orden de unos pocos minutos

AFS: Consistencia de la cache

- En AFS-1 antes de cada open se interrogaba al Vice con la marca de tiempo para ver la validez del fichero
 - Esto produce mucha comunicación cuando se escala el servicio
- En AFS-2 y siguientes se emplea la *promesa de devolución de llamada*
 - Ahorra la mayor parte de las comunicaciones
 - Obliga al Vice a mantener algo del estado de los clientes Venus a diferencia de NFS y AFS-1
 - Una lista de procesos Venus con *promesa de devolución de llamada*
 - Esta lista debe mantenerse en el disco de los servidores para evitar pérdidas en paradas

AFS: Consistencia de la cache

- Interfaz RPC de los servidores AFS para operaciones sobre ficheros

<i>Fetch(ida) → atrib, datos</i>	Devuelve los atributos (estado) y, opcionalmente, los contenidos del archivo identificado por el <i>ida</i> y registra una promesa de devolución de llamada sobre él.
<i>Store(ida, atrib, datos)</i>	Actualiza los atributos y (opcionalmente) los contenidos de un archivo especificado.
<i>Create() → ida</i>	Crea un nuevo archivo y registra una promesa de devolución de llamada sobre él.
<i>Remove(ida)</i>	Elimina el archivo especificado.
<i>SetLock(ida, modo)</i>	Establece un bloqueo sobre el archivo o directorio especificado. El modo del bloqueo puede ser compartido o exclusivo. Los bloqueos no eliminados expiran a los 30 minutos.
<i>ReleaseLock(ida)</i>	Desbloquea el archivo o directorio especificado.
<i>RemoveCallback(ida)</i>	Informa al servidor de que un proceso Venus ha volcado un archivo desde su caché.
<i>BreakCallback(ida)</i>	Esta llamada se realiza desde un servidor Vice a un proceso Venus. Cancela la promesa de devolución de llamada del archivo en cuestión.

AFS: Consistencia de la cache

- Si se produce una operación `open` y se tiene una copia en la cache y no ha transcurrido un tiempo `T` se entiende válida la copia
 - Puede haberse perdido alguna *devolución de llamada* por fallo de comunicación
 - En muchos sistemas `T` suele ponerse a 10 minutos
- Si varios clientes hacen un `open` y modifican su copia, esta no se modifica en el servidor hasta que no hacen un `close` del fichero
 - Todas las modificaciones menos las del último `close` se perderán
 - Los clientes deben implementar el control de concurrencia independientemente si lo precisan
- Este tipo de inconsistencias en principio no plantea problemas para la mayoría de las aplicaciones UNIX

AFS: Otros aspectos

- **Modificaciones del núcleo UNIX del servidor**
 - Debe estar modificado para permitir la utilización de apuntadores de ficheros en lugar de descriptores de ficheros
- **Bases de datos de ubicación**
 - Cada servidor contiene una copia de una base de datos que relaciona volumen y servidor que lo contiene.
 - Pueden ocurrir inconsistencias temporales cuando se traslada un volumen, pero como este se deja en el origen, no supone un problema
- **Hilos**
 - Venus y Vice utilizan hilos concurrentes para procesar solicitudes paralelas
- **Réplicas de sólo lectura**
 - /bin /usr/bin /man pueden estar replicados como volúmenes de sólo lectura en varios servidores
 - Se selecciona el servidor en función de la carga y accesibilidad
 - Cuando se hacen replicas sólo una copia es de lectura/escritura

AFS: Otros aspectos

- Transferencias al por mayor

- AFS transfiere trozos de 64K-bytes lo que optimiza el uso de la red

- Cacheado parcial de ficheros

- La necesidad de traer el fichero completo al cliente supone ineficiencia si sólo se necesita utilizar un trozo
- AFS-3 permite transferir trozos de ficheros de 64K-bytes que serán almacenados en la caché manteniendo su consistencia de AFS

- Prestaciones

- Los AFS benchmark ponen de manifiesto la disminución de la carga de los servidores de AFS con un modelo de *devolución de llamada* para notificar a los clientes la modificación de los archivos, frente a los NFS basado en un mecanismo de *timeout* para la validación de las páginas de la caché de los clientes

- Soporte de área amplia

- AFS-3 soporta múltiples celdas administrativas cada una con sus propios servidores, clientes, administradores del sistema y usuarios
- Cada celda es autónoma pero una *federación de celdas* puede cooperar en un espacio común

AFS: Avances recientes

■ Obtención de la semántica de actualización de una copia

- NFS no garantiza que los resultados de la escritura concurrente de varios clientes en un mismo fichero sea la misma que un sistemas UNIX cuando varios procesos escriben en un mismo fichero
- Spritely NFS
 - Añade a NFS las llamadas `open` y `close`
 - Cliente envía `open` a Servidor especificando modo (lectura, escritura o ambas) y número de procesos locales que tienen el fichero abierto para lectura y escritura
 - Cliente envía `close` a Servidor con contadores actualizados
 - Servidor mantiene una *tabla de archivos abiertos* con la IP y puerto de los clientes y sus contadores
 - Si un Cliente emite un `open` de lectura el Servidor consulta la tabla y si hay Clientes escribiendo hace una devolución de llamada al cliente para que deje de escribir en la caché y utilice el modo de *escritura a través* directamente sobre el servidor.
 - Si un Cliente emite un `open` de escritura se consulta en la tabla y todos los Clientes modifican su política de cache para evitar inconsistencias
 - La tabla del servidor se mantiene en disco para evitar pérdidas por caída, pero también se consulta a los clientes de tarde en tarde sobre los archivos abiertos
 - Este sistemas no sobrecarga demasiado las comunicaciones, obliga a mantener cierto estado de los clientes, complica la implementación y necesita de un sistema de recuperación de estado después de caída del servidor pero a cambio garantiza la semántica de una copia de UNIX

AFS: Avances recientes

- NQNFS (Not Quite NFS)

- Mantiene una tabla similar a Spritely NFS
- Utiliza un sistema de concesiones
 - Cuando un cliente solicita un fichero se le hace una concesión por un intervalo de tiempo pasado el cual tiene que renovar
- Se hace una devolución de llamada a todos los clientes que mantienen un fichero abierto cuando un cliente solicita abrir ese fichero de escritura, pero si el cliente no contesta se espera a que pase su concesión para conceder la solicitud de escritura
- Utiliza concesiones para ayudar a restablecer la información de los servidores después de la caída

AFS: Avances recientes

■ WebNFS

- Muchas aplicaciones Web podrían aprovecharse del acceso directo a servidores NFS sin las sobrecargas asociadas al SO UNIX de los clientes.
- Se permite a aplicaciones Web acceder a ficheros NFS públicos usando un apuntador de fichero público.
 - Así se evita el montado y el servicio de enlace de puertos (portmapper)
- Los cliente WebNFS interaccionan con un servidor NFS en el número de puerto 2049
- El cliente lanza un lookup utilizando un apuntador público de fichero
- El apuntador es interpretado por el VFS del servidor y se proporciona la información al cliente
- WebNFS permite hacer escrituras
- Proporciona un control de acceso y autenticación
 - Pero se puede deshabilitar ya que en muchos casos el cliente sólo solicita lectura de ficheros públicos
- Para leer una porción de un fichero de un servidor NFS que soporta WebNFS se precisa:
 - Una conexión TCP y
 - Dos llamadas RCP: un lookup y un read

AFS: Avances recientes

- WebNFS

- Ejemplo:

- Un servicio meteorológico publica una gran base de datos que se actualiza frecuentemente

`nfs://data.weather.gov/weatherdata/global.data`

- Un cliente que muestra mapas de tiempo podría acceder a la base de datos a través de la biblioteca de procedimiento WebNFS leyendo sólo los datos que le interesan.
 - Si se utilizara http habría que bajar toda la base de datos o acceder a una aplicación especial que sirviera sólo los datos necesarios.

AFS: Avances recientes

- NFS versión 4 (NFSv4)

- Incluye seguridad Kerberos,
- Trabaja con cortafuegos y proxis
- Permite listas de control de acceso o ACLs (del inglés, access control list)
- Utiliza operaciones con descripción del estado.
- Orientado a WAN
 - Incluye las características de WebNFS.
 - Más eficiente
 - Mejor escalabilidad
- NFS versión 4.1 (RFC 5661, January 2010) orientado a cluster de servidores y que permite acceso (pNFS paralelo pNFS)

AFS: Avances recientes

- Mejoras en la organización del almacenamiento
 - Redundant Arrays of Inexpensive Disks (RAID, Cadenas Redundantes de Discos Baratos)
 - Aumentan la velocidad por su acceso concurrente a diferentes bloques
 - Resistentes a fallos
 - Códigos detectores y correctores de errores
 - Log-structured file storage (LFS, Almacen de Archivo Estructurado en Histórico)
 - Al aumentar la cache se aumentaban las prestaciones de las lecturas pero no de las escrituras debido a que había de modificar bloque individuales de datos en el disco
 - Para aumentar el rendimiento de las escrituras LFS acumula un conjunto de escrituras en memoria y las realiza a disco en segmentos de tamaño fijo, grandes y contiguos.
 - Así los bloques se guardan en el orden que fueron actualizados