CC5303 – Sistemas Distribuidos

8.- Sistemas de Archivo Distribuido

Sebastián Blasco V.

Sistemas de Archivo Distribuido

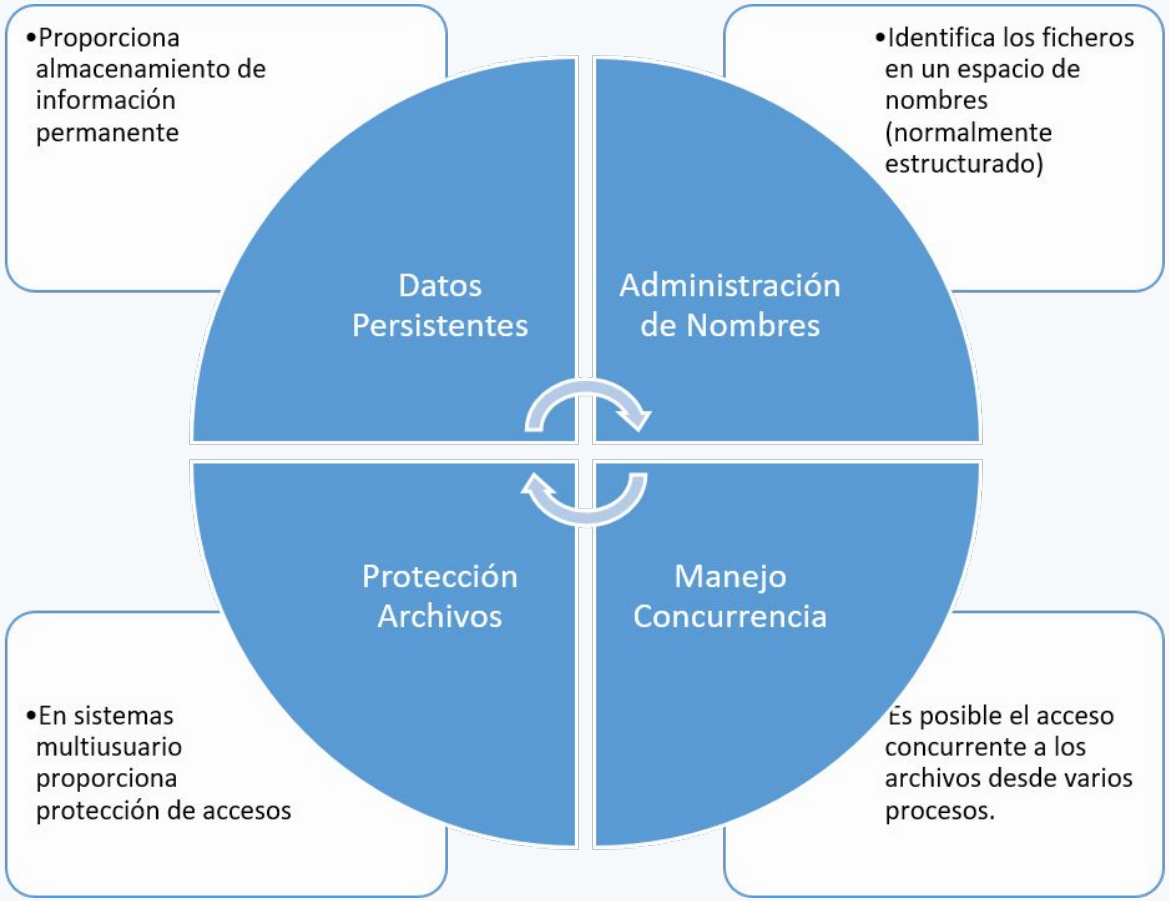
Los sistemas de archivo distribuidos permiten que múltiples procesos **compartan datos durante largos períodos en forma segura y confiable**.

Son la forma más común de SSDD

- Idea general
  - Dado una colección de discos conectados a un nodo, compartirlos como si estuvieran conectados a cada nodo

Sistemas de Archivo Distribuido

- Propiedades generales



Sistemas de Archivo Distribuido

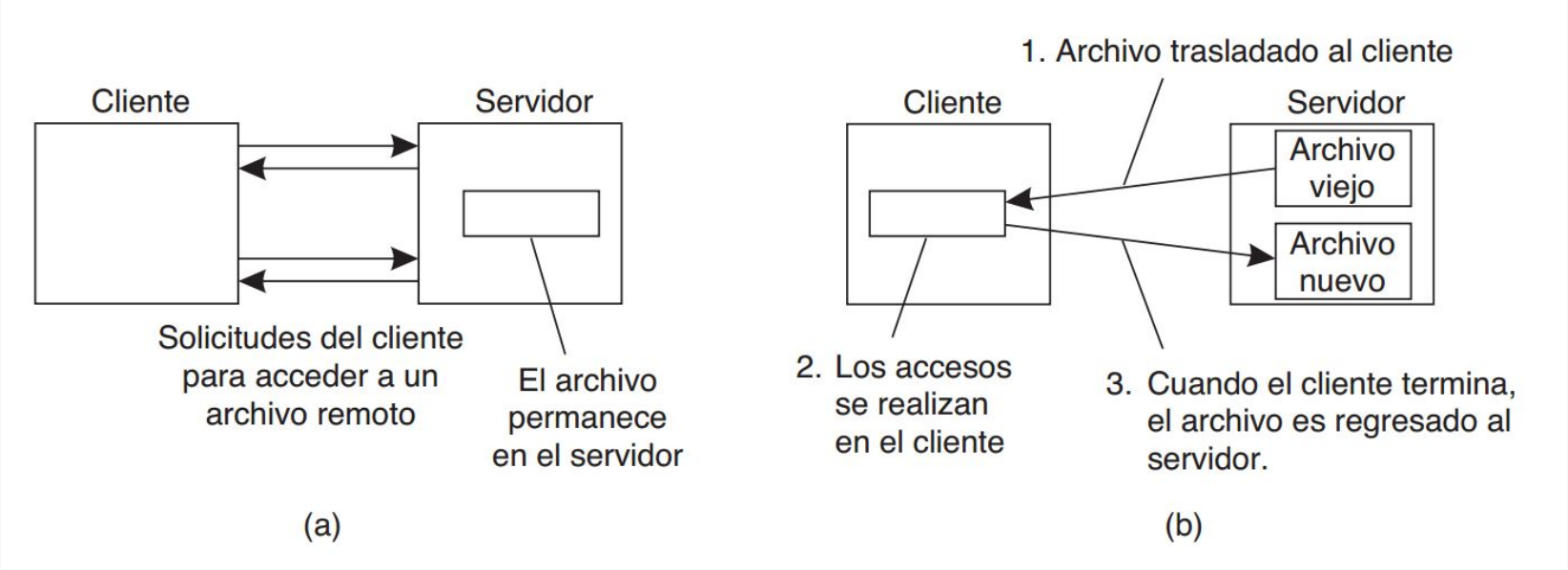
- Premisas a aceptar
  - La mayoría de los ficheros son de pequeño tamaño.
    - El fichero puede ser la unidad de recuperación.
  - La escritura es poco frecuente.
    - Esto alienta el caching y la replicación.
  - La compartición es poco frecuente.
    - La mayoría de los ficheros se acceden por
      - un lector y/o un escritor
      - algunos se acceden por *n* lectores y un escritor, y
      - muy rara vez se acceden por *n* lectores y *m* escritores
    - Por lo tanto, puede ser rentable una gestión optimista del caching y la replicación, que suponga que hay un único escritor y rectifique en caso de detectar a posteriori escrituras simultáneas.

Sistemas de Archivo Distribuido

- Premisas a aceptar
  - El acceso suele ser secuencial y existe un alto grado de localidad.
    - Esto promueve el buffering para proporcionar anticipación en los accesos.
  - La mayoría de los ficheros tienen una vida muy corta (por ejemplo, ficheros temporales). Hay que tender a gestionarlos localmente.
  - Existen clases de ficheros, con propiedades diferenciadas (por ejemplo ejecutables, que rara vez se modifican).

Sistemas de Archivo Distribuido

- Dos enfoques principalmente
  - Modelo de acceso remoto
  - Modelo de carga y descarga

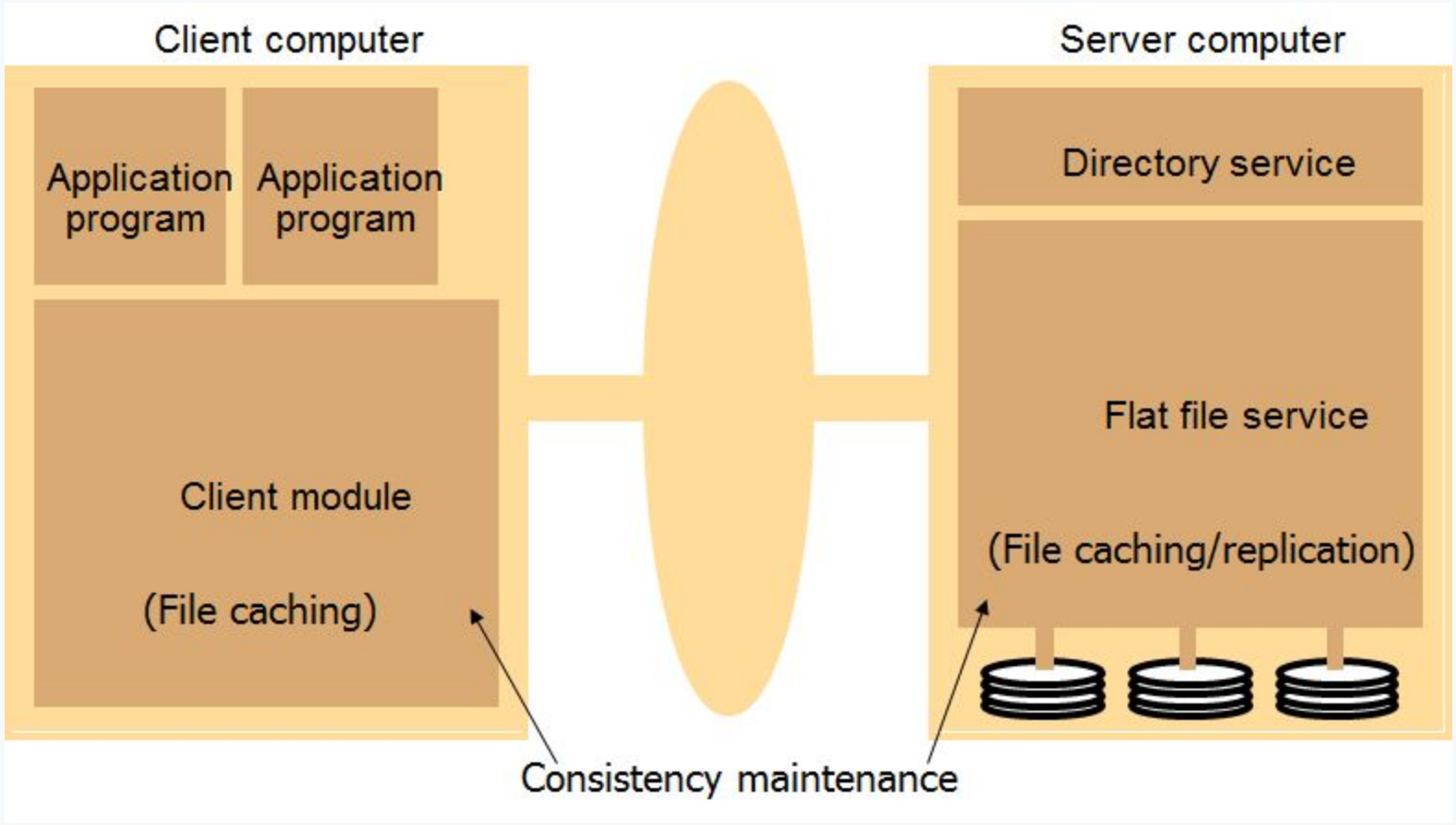


(a) Modelo de acceso remoto. Caso NFS

(b) Modelo de carga y descarga. Caso FTP

Sistemas de Archivo Distribuido

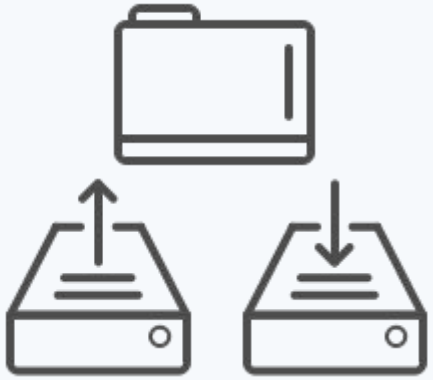
NFS, idea general



Arquitectura DFS

Basado en esquema Cliente-Servidor

- Cliente
- Servicio de nombres (directorios)
- Servicio de ficheros



Arquitectura DFS

Basado en esquema Cliente-Servidor

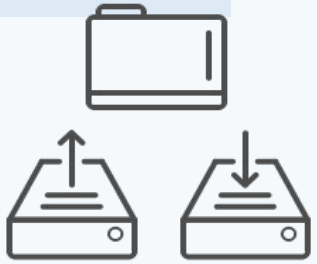
- Cliente
  - Interfaz local con la aplicación.
  - Interpreta las llamadas al sistema sobre ficheros y genera las peticiones (habitualmente RPCs) para los accesos remotos.
  - Conoce la ubicación de los servicios de nombres y de ficheros.
  - Gestiona el almacenamiento local (caching).

Arquitectura DFS

Basado en esquema Cliente-Servidor

- Servicio de nombres (directorios)
  - Es el encargado de proporcionar transparencia en la ubicación.
  - En general, es una base de datos con elementos (nombre, UFID).
  - Algunos atributos del fichero se mantienen por el servicio de nombres
    - Tipo de fichero (ordinario o directorio), propietario, permisos, etc.

Arquitectura DFS



Basado en esquema Cliente-Servidor

- Servicio de ficheros
  - Mantiene el contenido de los ficheros (y directorios) y las propiedades de los ficheros
    - Tiempos de creación, último acceso y última modificación, etc.
  - Un fichero se identifica en el servicio de ficheros mediante un identificador único de fichero.
  - La interfaz del servicio de ficheros con el cliente ofrece operaciones como leer, escribir, crear, borrar, etc.

Sistemas de Archivo Distribuido

- Problemas a enfrentar
  - **Naming & transparencia**
  - Acceso Remoto de Archivos (Servidores de Fichero)
  - Tipo de Servidores
  - Caching
  - Consistencia por replicación

Naming & Transparencia

- Transparencia de Localización
  - El nombre de archivo no revela la localización física del almacenamiento del archivo
  - Muchos lo tienen
- Independencia de Localización
  - El nombre de archivo no cambia si la localización de su almacenamiento cambia.
  - La gran mayoría lo tiene

- Estratégias
  - Nombres absolutos
  - Puntos de Montaje

Naming & Transparencia

Nombres Absolutos

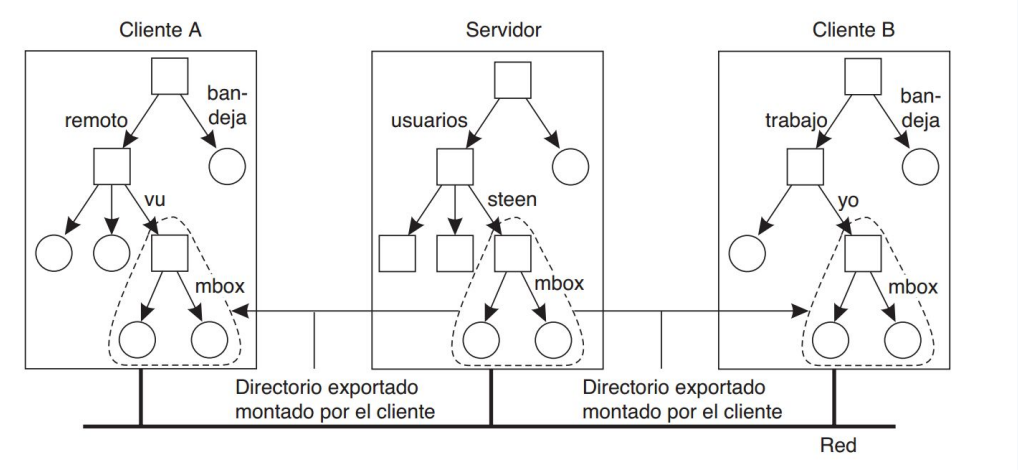
<machine name, pathname>
Ej: AppleShare, Windows NT

- Ventajas
  - Fácil de encontrar nombre de archivo (completamente especificado)
  - Fácil de agregar y eliminar nuevos nombres
  - Ningún estado global
  - Escala fácilmente
- Desventajas
  - El usuario debe conocer el nombre completo - consciente de qué archivos son locales y que son remotos
  - El archivo depende de la ubicación (no se puede mover)
  - Hace más difícil compartir
  - No tolerante a fallos

Naming & Transparencia

Puntos de Montaje

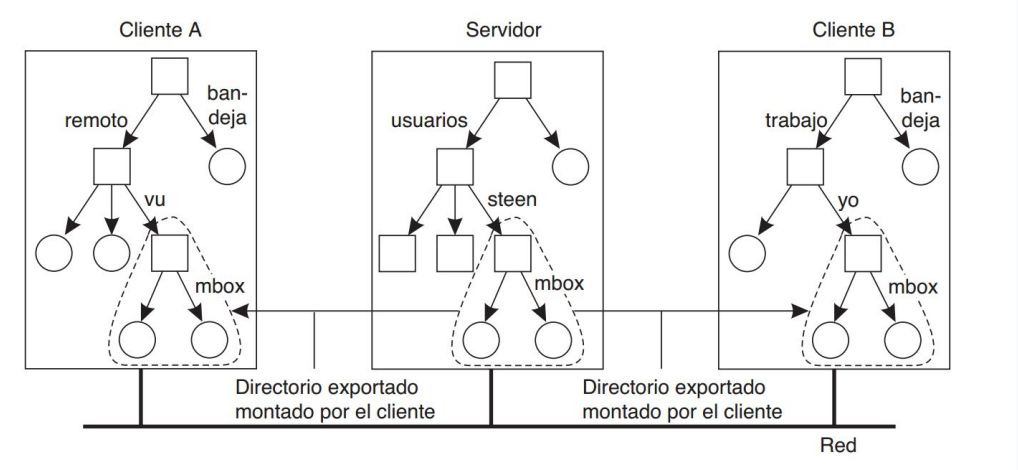
- Los nombres están organizados en un espacio de nombre jerárquico.
- La transparencia se logra permitiendo que el cliente sea capaz de montar un sistema de archivo remoto en su propio sistema de archivo local.
- NFS permite que los clientes monten sólo una parte del sistema de archivos.



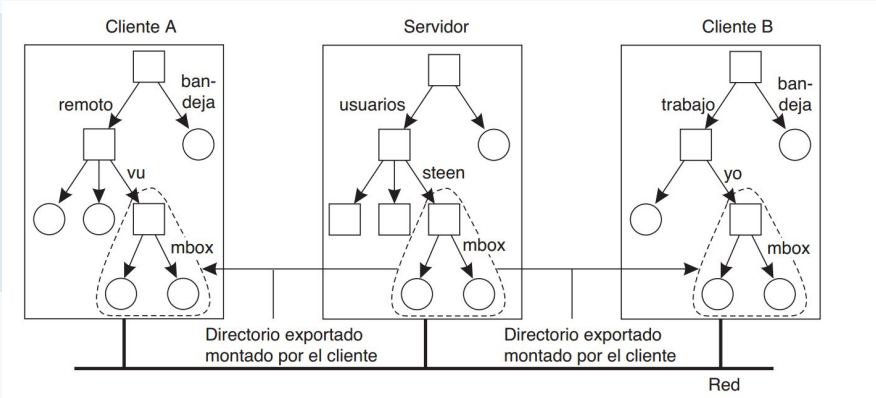
Naming & Transparencia

Puntos de Montaje

- Ventajas:
  - Ubicación transparente
  - El nombre remoto puede cambiar a través de los reinicios
- Desventajas:
  - Una sola estrategia unificada difícil de mantener
  - El mismo archivo puede tener nombres diferentes



Naming & Transparencia



Puntos de Montaje

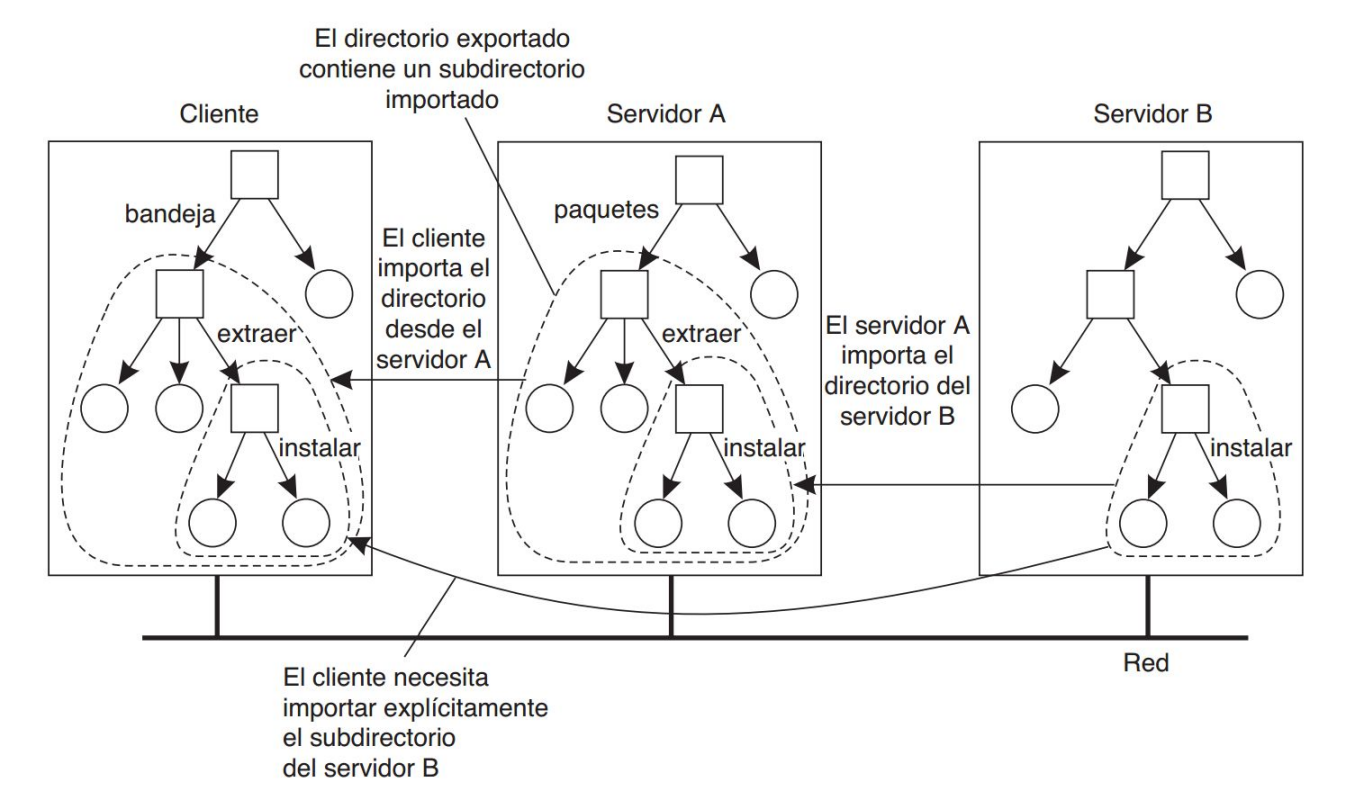
Este método de diseño tiene una seria implicación: **En principio, los usuarios no comparten espacios de nombre**.

- El archivo de nombre /remote/vu/mbox en el cliente A se llama /work/me/mbox en el cliente B.
- Un nombre de archivo depende, por consiguiente, de cómo organizan los clientes su propio espacio de nombre local, y en dónde se monten los directorios exportados.
- La desventaja de usar este método en un sistema de archivo distribuido es que compartir archivos se vuelve mucho más difícil.

Naming & Transparencia

Puntos de Montaje

- Montaje Recursivo



Sistemas de Archivo Distribuido

- Técnicas de distribución de archivos
  - Un archivo se distribuye a través de múltiples servidores. La idea básica es simple: distribuyendo un archivo grande entre múltiples servidores, es posible buscar sus diferentes partes en paralelo.

Sistemas de Archivo Distribuido

- Problemas a enfrentar
  - Naming & transparencia
  - **Acceso Remoto de Archivos (Servidores de Fichero)**
  - Tipo de Servidores
  - Caching
  - Consistencia por replicación

Servidores de ficheros

- ¿Cómo tratar el **acceso remoto de archivos**?
- El objetivo en el diseño de los servidores de ficheros distribuidos es el proporcionar una semántica lo más cercana posible a la que ofrecen los sistemas centralizados.
  - Ello sin degradar significativamente el rendimiento general del acceso.
  - ¡Cuidado con la latencia!

- Se utiliza extensamente el caching y se utilizan mecanismos de gestión de las copias que permiten **compromisos razonables entre semántica y rendimiento**.

Servidores de ficheros

Semánticas de compartición

- Compromiso entre el rendimiento deseado y la semántica que queremos garantizar en el sistema de ficheros distribuido cuando se producen accesos concurrentes.

- Podemos distinguir varios tipos de semánticas:
  - Semántica UNIX.
  - Semántica de sesión.
  - Ficheros inmutables.
  - Semántica de transacciones.

Servidores de ficheros

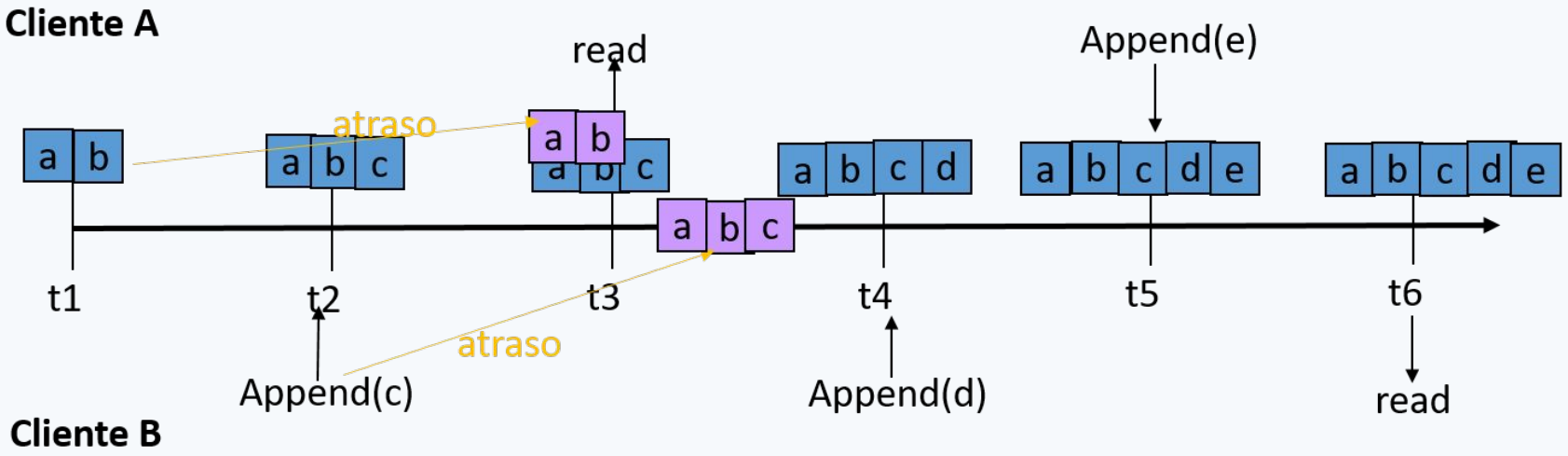
Semánticas de compartición

- Semántica UNIX.
  - Caracteriza el acceso a los ficheros de los sistemas UNIX clásicos en entornos centralizados.
  - Las operaciones sobre un fichero se ordenan totalmente en el tiempo
    - Una lectura devuelve la última actualización del fichero.
    - Ordenación absoluta (Como si sólo existiera una sola copia y con actualización instantánea)
  - Problema
    - Es complejo para un sistema distribuido proporcionar semántica UNIX.
    - **Retrasos en la red**

Servidores de ficheros

Semánticas de compartición

- Semántica UNIX.



Ordenación absoluta (Como si sólo existiera una sola copia y con actualización instantánea)

Retrasos en la red

Servidores de ficheros

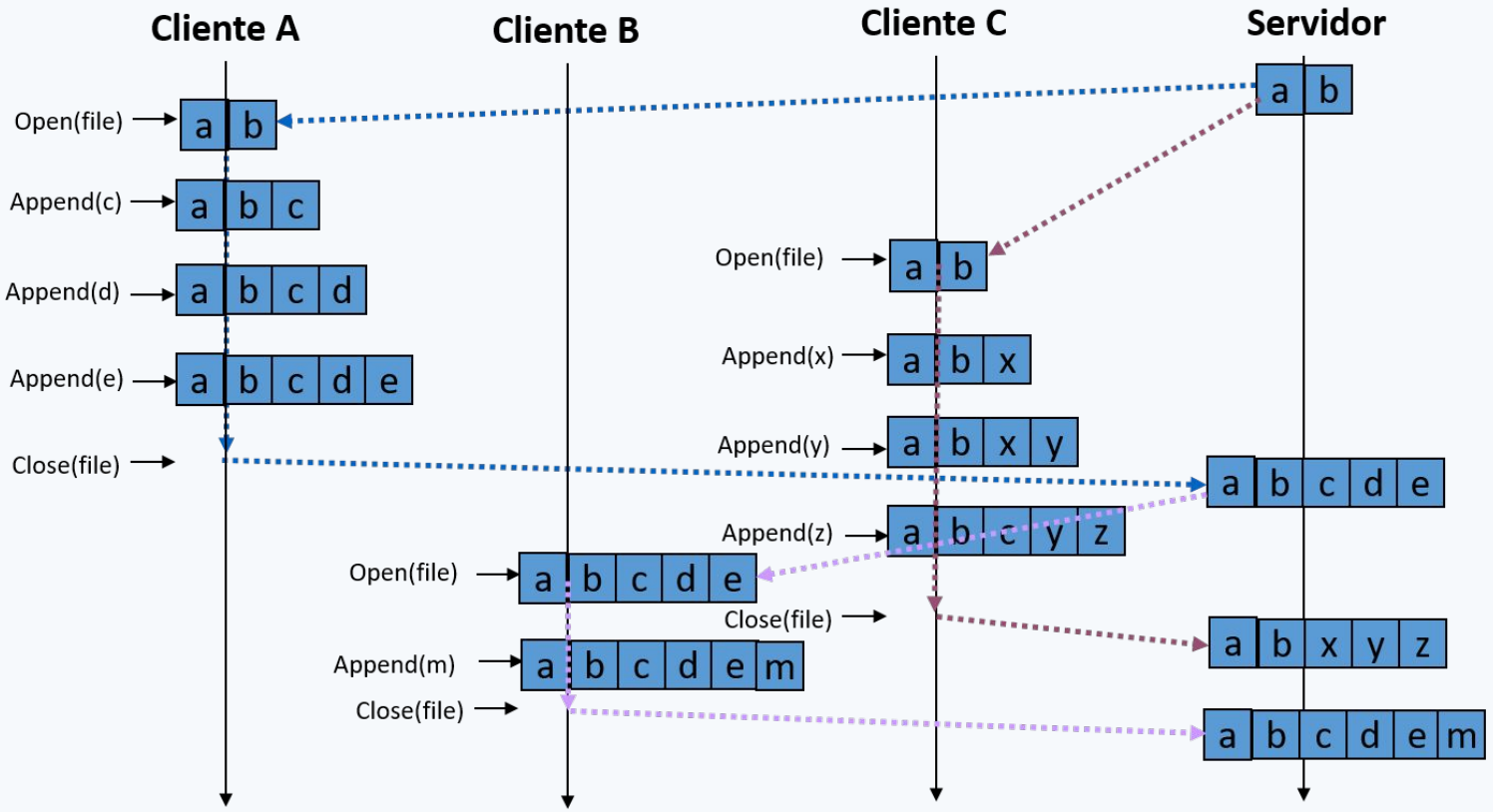
Semánticas de compartición

- Semántica de Sesión.
  - En una sesión de uso del fichero (desde que éste se abre hasta que se cierra) el proceso ve una copia privada del fichero;
    - No se comparte nada del estado modificado del fichero (por ejemplo el apuntador a la posición actual).
  - Equivale a trabajar sobre una copia local que se carga cuando el fichero se abre y se actualiza en el servidor cuando se cierra.
  - Problema:
    - Condiciones de carrera cuando se escribe en el servidor la copia actualizada, **cuya resolución compete al usuario o a la aplicación**.
      - Las escrituras de archivos pueden sobrescribir las actualizaciones anteriores.
      - El bloqueo de archivos es necesario para evitar que se sobreescriba

Servidores de ficheros

Semánticas de compartición

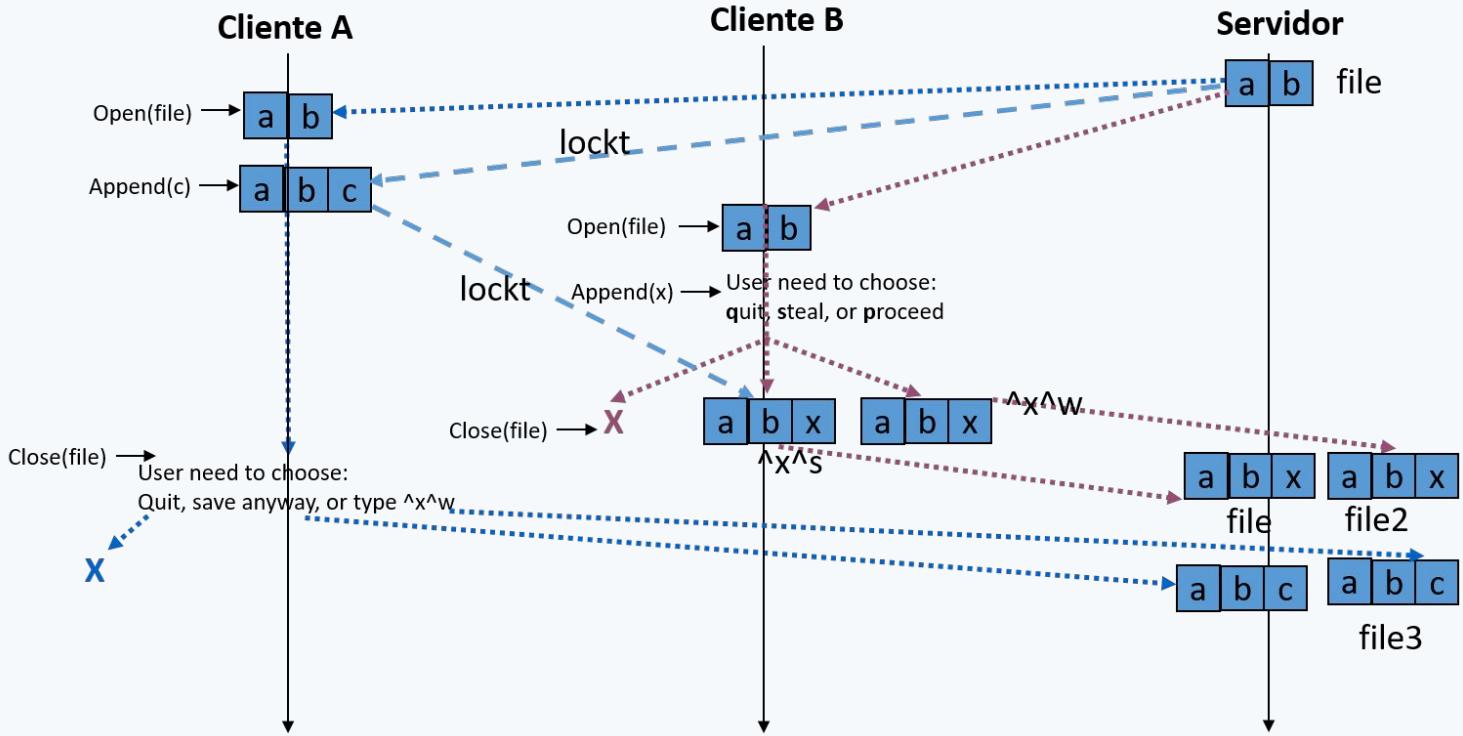
- Semántica de Sesión.



Servidores de ficheros

Semánticas de compartición

- Semántica de Sesión.
  - Con bloqueo de archivos



Servidores de ficheros

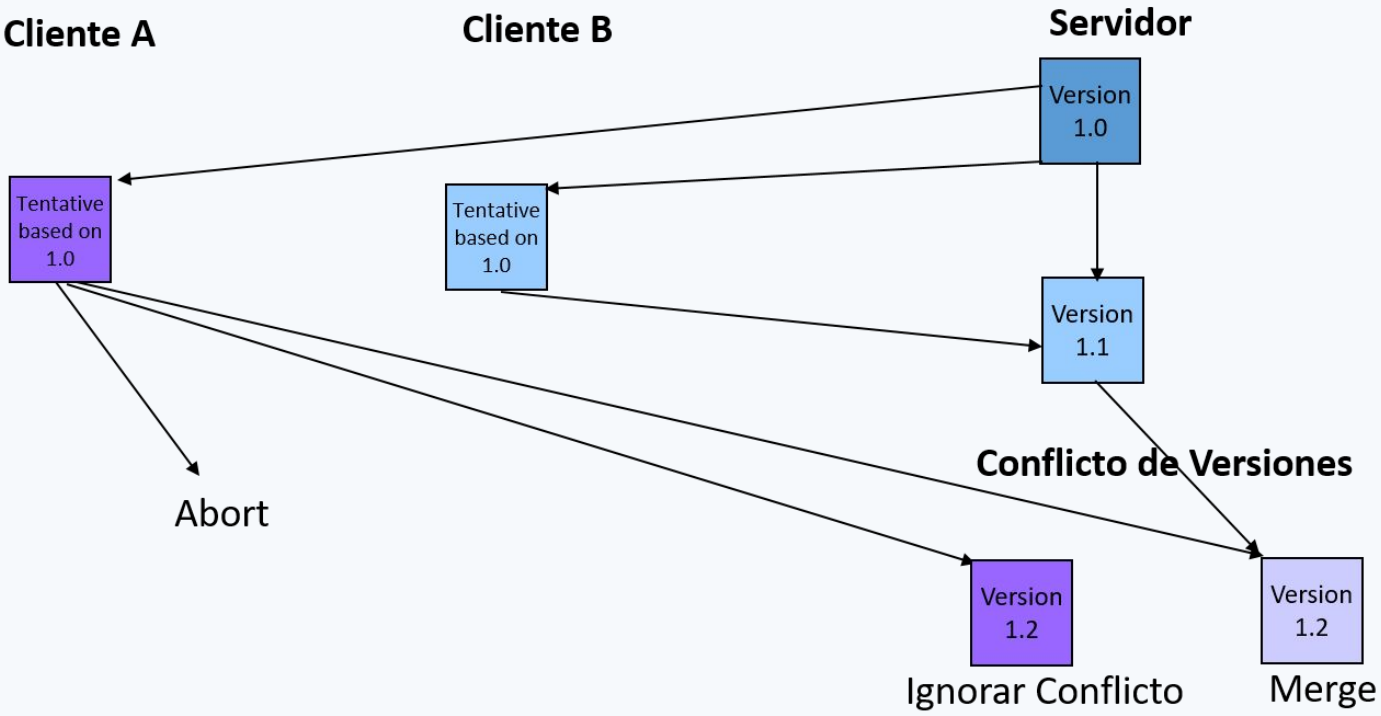
Semánticas de compartición

- Ficheros inmutables.
  - Los ficheros no se modifican, sino que se reemplazan atómicamente (los directorios sí se modifican) por nuevas versiones.
  - Es posible la compartición concurrente para lectura, pero si se pretende acceder un fichero abierto por otro proceso para escritura existen dos alternativas:
    - (a) Considerar error la operación de abrir, y
    - (b) Obtener la versión anterior del fichero.
  - Esta semántica es adecuada en servicios particulares, como los de back-up y los repositorios de información con gestión de versiones (por ejemplo ¿?).

Servidores de ficheros

Semánticas de compartición

- Ficheros inmutables.



Servidores de ficheros

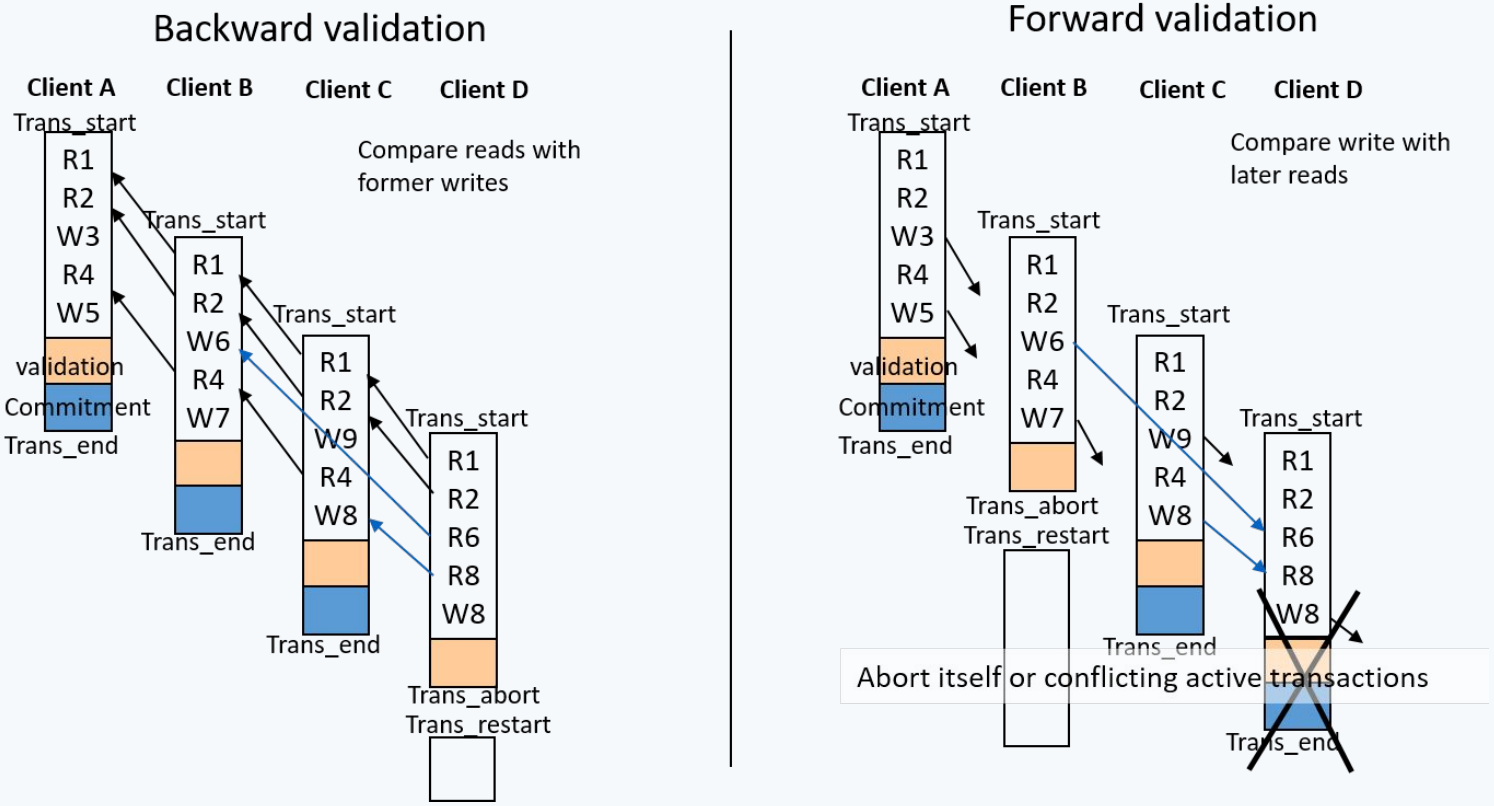
Semánticas de compartición

- Semántica de transacciones.
  - El uso de transacciones permite definir explícitamente secuencias de operaciones sobre ficheros.
  - Garantiza la semántica transaccional definida por las propiedades ACID.

Servidores de ficheros

Semánticas de compartición

- Semántica de transacciones.



Sistemas de Archivo Distribuido

- Problemas a enfrentar
  - Naming & transparencia
  - Acceso Remoto de Archivos (Servidores de Fichero)
  - **Tipo de Servidores**
  - Caching
  - Consistencia por replicación

Tipo de Servidor

Dependiendo de la información que almacena el servidor acerca del fichero que está siendo accedido por un cliente, se pueden distinguir dos categorías de servidores, que determinarán estrechamente las características del servicio.

- Servidor sin estado.
- Servidor con estado.

Tipo de Servidor

- Servidor sin estado.
  - El servidor no almacena información del cliente en una sesión sobre un fichero.
  - Las interfaces cliente-servidor que ofrece este tipo de sistema no incluye primitivas específicas de abrir/cerrar. **El cliente suministra en cada llamada toda la información necesaria para realizar la operación** (ej. el puntero a la posición de acceso actual).
  - Ventajas
    - El fallo de un cliente no afecta en absoluto al servidor.
  - Contra
    - Es difícil el proporcionar semántica UNIX.

Tipo de Servidor

- Servidor con estado.
  - El servidor crea una entrada para el fichero ante una invocación de abrir fichero de un cliente.
    - Las sucesivas invocaciones de acceso al fichero requieren mensajes más cortos, ya que el servidor almacena el apuntador a la última posición accedida y otra información, que puede incluir hasta bloques del fichero, lo que posibilita lectura anticipada en el servidor.
  - **Este enfoque limita de forma inherente el número de ficheros abiertos simultáneamente**.
  - Problema
    - El servidor tiene que gestionar el posible fallo de los clientes, para liberar la información de estado de los ficheros abiertos por el cliente que falla, evitando así la degradación del servidor.

Sistemas de Archivo Distribuido

- Problemas a enfrentar
  - Naming & transparencia
  - Acceso Remoto de Archivos (Servidores de Fichero)
  - Tipo de Servidores
  - **Caching**
  - Consistencia por replicación

Caching

- Mecanismo de almacenamiento temporal de (trozos de) ficheros en el nodo cliente.
- Pretenden minimizar los costes de comunicación que el acceso remoto lleva asociados.
- La unidad de gestión, cantidad de información que se transmite en cada petición, puede ser el fichero completo o un bloque.
- Transmitir cantidades grandes de información proporciona lectura anticipada (buffering)
  - Potencia la localidad espacial.

Caching

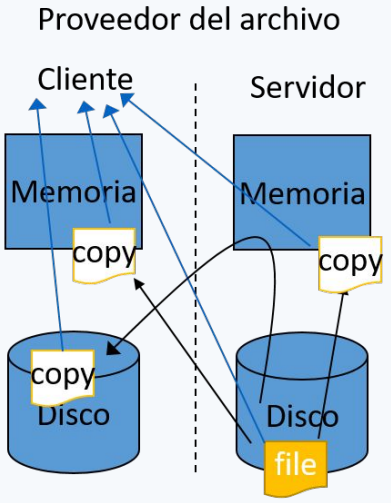
El caching se puede soportar bien en el disco local, de forma persistente, bien en memoria, no persistente.

- Disco local
  - Rápido tiempo de acceso*
  - Seguro si falla el nodo
  - Difícil mantener copia local coherente con copia remota
  - Requiere que el cliente tenga disco!
- Memoria local
  - Inmediato tiempo de acceso*
  - Funciona sin discos
  - Difícil mantener copia local coherente con copia remota
  - Tamaño de caché más pequeño
  - No tolerante a fallos

(*) en comparación con el control remoto

Caching

Locación Datos	Pro	Contra
Sin Cacheo	-Modificación directa	-Alto acceso a disco -Alto tráfico de red
En memoria del Servidor	-Bajo acceso a disco -Fácil Implementación -Semantica UNIX	-Alto tráfico de red
En disco del cliente	-Bajo acceso a la red -Sin restricción de tamaño de archivo	-Problemas consistencia caché -¿Semantica UNIX? -Alto acceso a disco -Necesita un disco
En memoria del cliente	-Máxima performance -No necesita disco -Escalable	-Restricción tamaño de archivos -Problemas consistencia caché -¿Semantica UNIX?



Sistemas de Archivo Distribuido

- Problemas a enfrentar
  - Naming & transparencia
  - Acceso Remoto de Archivos (Servidores de Fichero)
  - Tipo de Servidores
  - Caching
  - **Consistencia por replicación**

Consistencia por replicación

- La política de gestión de la consistencia condiciona la semántica de compartición.

- Las políticas básicas para gestionar la consistencia son las siguientes:
  - Write-through
  - Write-on-close
  - Gestión centralizada

Consistencia por replicación

- Write-through
  - Cuando un elemento se modifica, se copia en el servidor.
  - Se suelen introducir adaptaciones para mejorar el rendimiento:
    - Escritura retardada (**Delayed-write scheme**)
      - Acumula modificaciones y sincroniza a intervalos regulares
    - No copiar en el servidor los ficheros temporales.

Consistencia por replicación

- Write-on-close
  - El fichero se escribe en el servidor cuando se cierra.
  - Determina semántica de sesión.
  - Una mejora consiste en retardar la escritura
    - Es frecuente que un fichero se borre después de cerrar.

Consistencia por replicación

- Gestión centralizada
  - El servidor de ficheros gestiona la apertura/cierre de ficheros mediante un algoritmo de sincronización lectores-escritores.
  - Proporciona semántica UNIX
  - Es poco escalable y no tolerante a fallos

Consistencia por replicación

- Salvo en el caso de la gestión centralizada, las políticas de gestión de la consistencia descritas **no son suficientes para proporcionar la semántica definida** (en particular UNIX).

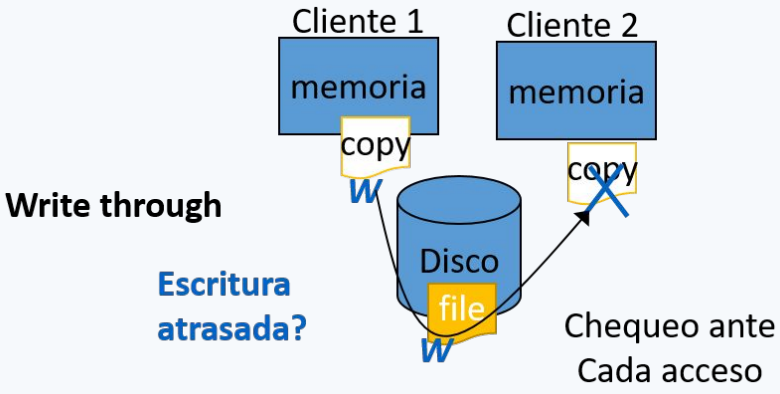
- Se requiere un mecanismo adicional de validación para **permitir a un nodo conocer cuándo la réplica de un fichero que está se usando en su caché ha quedado obsoleta por la actualización** de otra réplica del mismo fichero en otro nodo (no necesariamente el servidor).

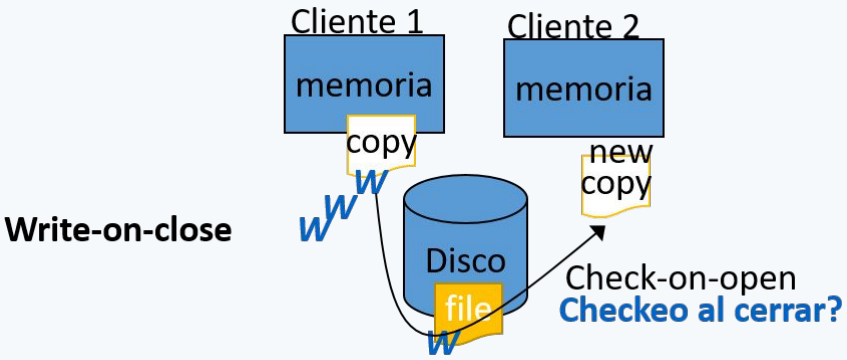
Consistencia por replicación

- Existen dos políticas básicas de validación, dependiendo de donde parta la iniciativa:
  - (a) **Desde los clientes**, accediendo periódicamente a los atributos del fichero en el servidor, para ver si se ha modificado.
    - El periodo entre validaciones es un parámetro crítico: preservar la semántica requiere periodos cortos, a costa de sobrecargar la red.
  - (b) **Desde el servidor**, notificando a los clientes cuando una copia ha quedado obsoleta (callback).
    - Requiere almacenar algo de estado en el servidor.

Consistencia por replicación

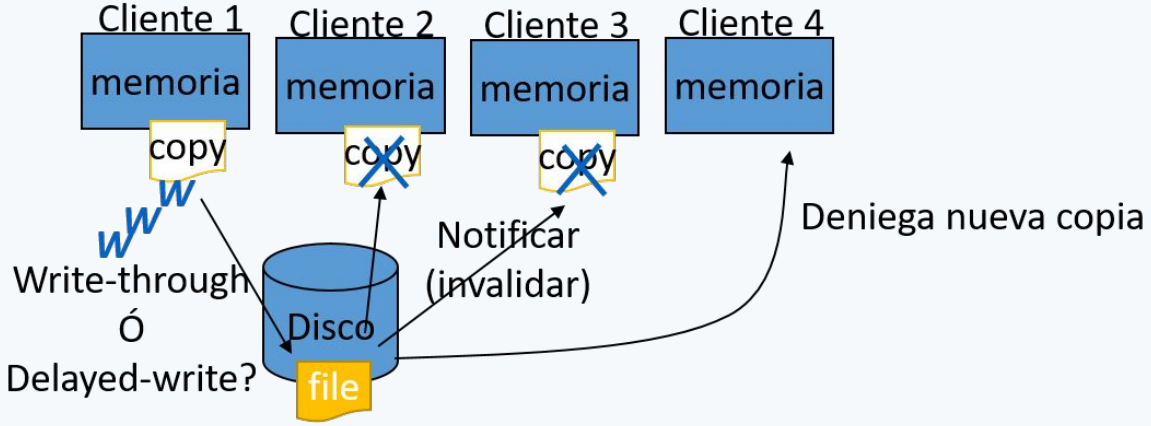
- Approach del cliente:
  - Comprobación antes de cada acceso (semántica similar a Unix pero demasiado lenta)
  - Comprobación periódica (mejor rendimiento pero semántica difusa de intercambio de archivos)
  - Comprobación en archivo abierto (simple, adecuado para semántica de sesión)
  - **Problema: Potencial alto tráfico de red**





Consistencia por replicación

- Approach del Servidor:
  - Hacer un seguimiento de los clientes que tienen una copia.
  - Denegar una nueva solicitud, ponerla en cola y deshabilitar la caché.
  - Notificar a todos los clientes de cualquier actualización del archivo original.
  - Problema:
    - **Viola el modelo cliente-servidor y requiere servidores con estado**



Fin