Sistemas Operativos

Archivos distribuidos

Armando Arce, Escuela de Computación, arce@itcr.ac.cr

Tecnológico de Costa Rica

Sistemas de archivos distribuidos

Un sistema de archivos distribuidos (DFS, distributed file system) es una implementación distribuida del modelo clásico de compartición de tiempo de un sistema de archivos, en el que múltiples usuarios comparten archivos y recursos de almacenamiento.

- El propósito de un sistema DFS es emplear el mismo tipo de compartición cuando los archivos están físicamente dispersos entre los sitios que componen un sistema distribuido.

Conceptos esenciales

Un sistema distribuido es una colección de computadoras débilmente acopladas interconectadas por una red de comunicaciones.

- Estas computadoras pueden compartir archivos físicamente dispersos utilizando un sistema de archivos distribuidos (DFS).

Conceptos esenciales

Un servicio es una entidad software que se ejecuta en una o más máquinas y que proporciona un tipo particular de función a los clientes.

- Un servidor es el software de servicio que se ejecuta en una única máquina.
- Un cliente es un proceso que puede invocar un servicio utilizando un conjunto de operaciones que forman su interfaz de cliente.

Conceptos esenciales

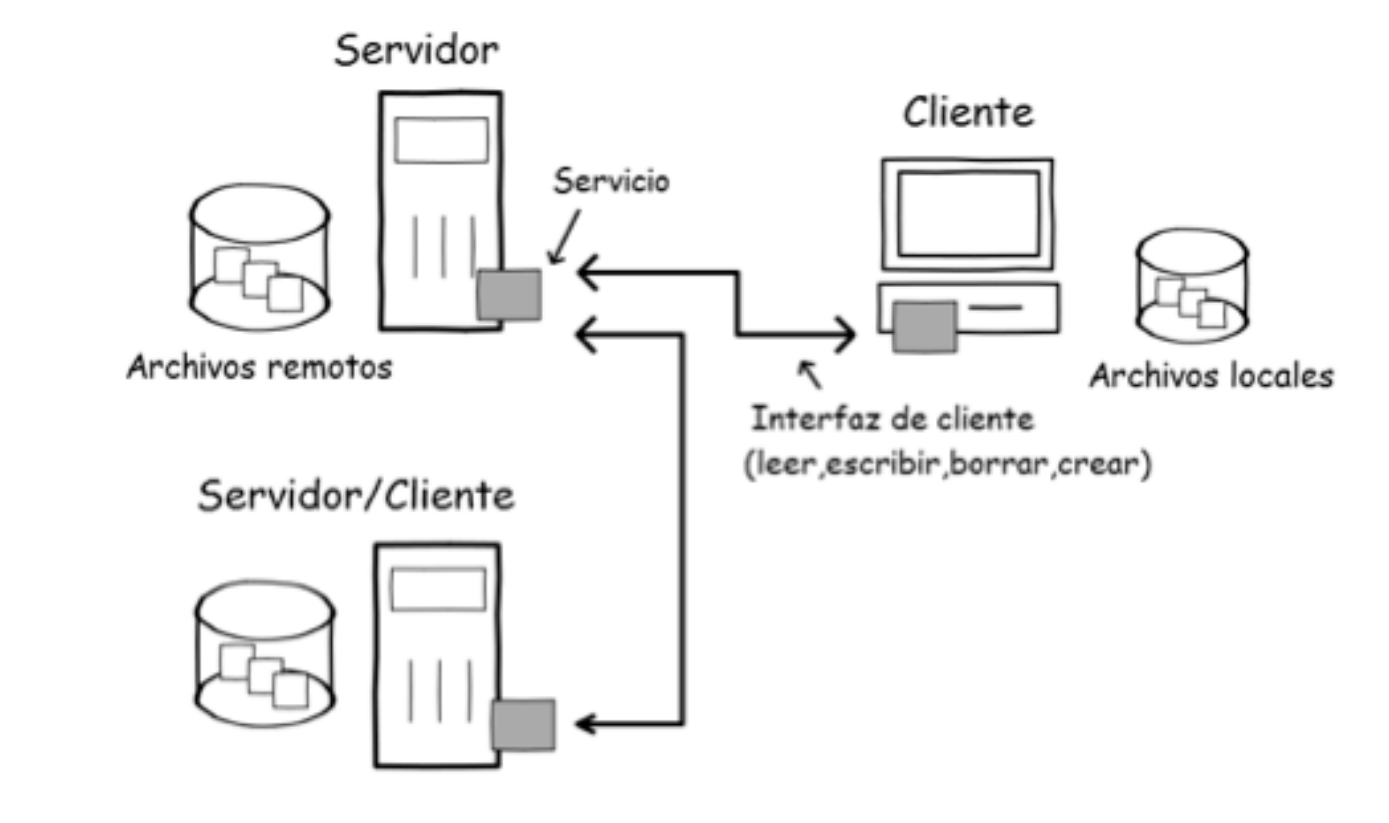

Figure 1:

Conceptos esenciales

Un sistema de archivos proporciona servicios de archivos a los clientes.

- Una interfaz de cliente para una servicio de archivos está formada por un conjunto de operaciones primitivas de archivos, como la creación de un archivo, la de borrado de un archivo, la de lectura de un archivo y la de escritura en un archivo.

Conceptos esenciales

Un DFS es un sistema de archivos cuyos clientes, servidores y dispositivos de almacenamiento están dispersos entre las máquinas de un sistema distribuido.

- En lugar de existir un único repositorio de datos centralizado, el sistema tiene frecuentemente dispositivos de almacenamiento múltiples e independientes.

Conceptos esenciales

Idealmente, un DFS debe aparecer a ojos de sus clientes como si fuera un sistema de archivos centralizado convencional.

- La multiplicidad y la dispersión de sus servidores y de sus dispositivos de almacenamiento deben ser invisibles para los usuarios.

Nombrado y transparencia

El nombrado es una correspondencia entre los objetos lógicos y físicos.

- Por ejemplo, los usuarios tratan con objetos lógicos de datos representados por nombres de archivos, mientras que el sistema manipula bloques físicos de datos almacenados en las pistas de los discos.
- En un DFS transparente, se añade una nueva dimensión a esta abstracción: la de ocultar en qué lugar de la red reside el archivo.

Nombrado y transparencia

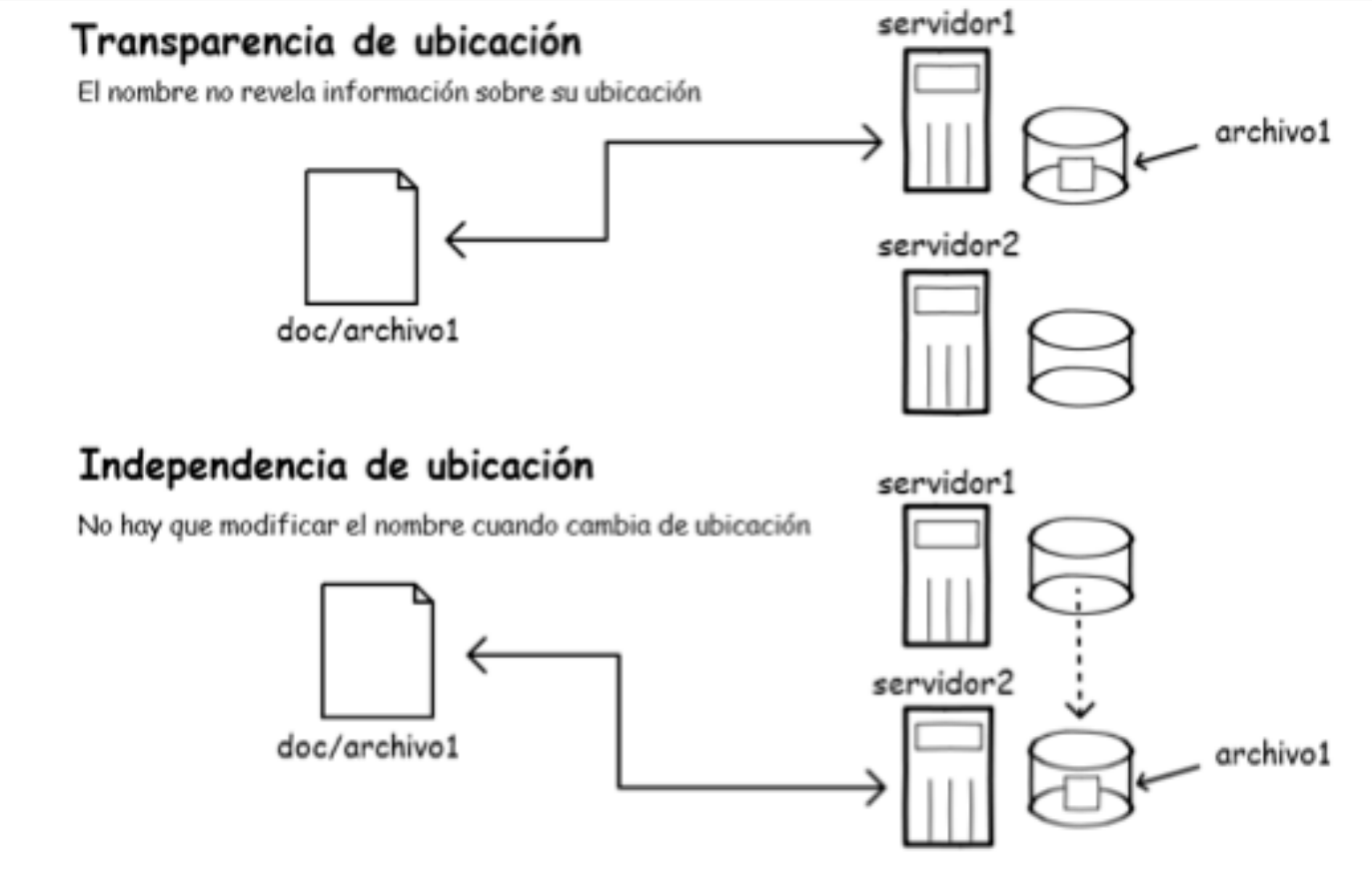

Figure 2:

Estructuras de nombrado

Es necesario diferenciar dos ideas relacionadas en lo que respecta a la correspondencias de nombres en un DFS:

1. Transparencia de ubicación: El nombre de un archivo no revela ninguna información acerca de la ubicación física de almacenamiento del archivo.
2. Independencia de ubicación: No es necesario modificar el nombre de un archivo cuajo varia la ubicación física de almacenamiento del archivo.

Estructuras de nombrado

La mayoría de los sistemas DFS actuales proporciona la transparencia respecto a la ubicación.

- Sin embargo, estos sistemas no soportan la migración de archivos; es decir, resulta imposible cambiar automáticamente la ubicación de un archivo.

Esquemas de nombrado

Existen tres técnicas principales para construir esquemas de nombrado en un DFS.

- Con el enfoque más simple, cada archivo se identifica mediante una combinación de su nombre de *host* y de su nombre local, lo que garantiza un nombre único en todo el sistema.
- En otros sistemas, por ejemplo, cada archivo se identifica unívocamente mediante el nombre *host:nombre-local*, donde *nombre-local* es una ruta estilo UNIX.

Esquemas de nombrado

La segunda técnica fue popularizada por el sistema de archivos de red (NFS, network file system) de Sun.

- NFS proporciona un medio para asociar directorios remotos a los directorios locales, proporcionando así la apariencia de un árbol de directorio coherente.
- Las primeras versiones de NFS sólo permitían acceder transparentemente a los directorio remotos previamente montados.

Esquemas de nombrado

Con la aparición de la característica de *automontaje*, el montaje se realiza según es necesario, basándose en una tabla de puntos de montaje y nombres de estructura de archivos.

- Los componentes se integran para soportar una compartición transparente, aunque esta integración está limitada y no es uniforme, porque cada máquina puede asociar directorios remotos distintos a su árbol.
- La estructura resultante es bastante versátil.

Esquemas de nombrado

Se puede conseguir una total integración de los sistemas de archivos componentes utilizando la tercera de las técnicas.

- Con ella, hay una única estructura global de nombres que abarca a todos los archivos del sistema.
- Idealmente, la estructura de sistemas de archivos compuesta es isomorfa a la estructura de un sistema de archivos convencional.

Acceso remoto a archivos

Una forma de llevar a cabo la transferencia de datos es mediante un mecanismo de servicio remoto, mediante el cual las solicitudes de acceso se entregan al servidor, la máquina servidora realiza los accesos y los resultados se devuelven al usuario.

- Una de las formas más comunes de implementar un servicio remoto es el paradigma de llamadas a procedimientos remotos (RPC, remote procedure call), del que ya se ha hablado anteriormente.
- Existe una analogía directa entre los métodos de acceso a disco en los sistemas de archivos convencionales y el método de servicio remoto en un DFS. Utilizar el método de servicio remoto es análogo a realizar un acceso a disco por cada solicitud de acceso.

Acceso remoto a archivos

Para garantizar un rendimiento razonable del mecanismo de servicio remoto, se puede utilizar algún tipo de caché.

- En los sistemas de archivos convencionales, la razón de utilizar el almacenamiento en caché es reducir la E/S de disco (incrementando así la velocidad), mientras que en un DFS, el objetivo es reducir tanto el tráfico red como el de E/S de disco.

Esquema básico de caché

El concepto de almacenamiento en caché es simple.

- Si los datos necesarios para satisfacer la solicitud de acceso no se encuentran ya en la caché, entonces se trae una copia de dichos datos desde el servidor hasta el sistema cliente.
- Los accesos se realizan sobre la copia almacenada en caché. La idea es retener en la caché los bloques de disco a los que se ha accedido recientemente, de modo que los accesos repetidos a la misma información puedan gestionarse localmente, sin necesidad de tráfico red adicional.

Esquema básico de caché

La implementación de algún tipo de sustitución (por ejemplo, la de los datos menos recientemente utilizados) hace que el tamaño de la caché se mantenga acotado.

- No existe correspondencia directa entre los accesos y el tráfico dirigido al servidor.
- Los archivos pueden seguir identificándose con una copia maestra que reside en la máquina servidora, pero una serie de copias del archivo (o de partes del mismo) estarán dispersas en las diferentes cachés.

Esquema básico de caché

Cuando se modifica una copia almacenada en caché, será necesario reflejar los cambios en la copia maestra, con el fin de preservar la correspondiente semántica de coherencia.

- El problema de mantener las copias de caché coherentes con el archivo maestro se denomina problema de la coherencia de caché, y se hablara de dicho problema más adelante.

Granularidad de datos

La granularidad de los datos almacenados en caché en un DFS puede variar, pudiendo definirse esa granularidad en el nivel de bloque de archivo o en el de un archivo completo.

- Usualmente, se almacenan en caché más datos de los necesarios para satisfacer un único acceso, de modo que se pueda dar servicio a muchos accesos mediante los datos almacenados en caché.

Granularidad de datos

Este procedimiento se parece bastante al mecanismo de lectura anticipada de disco.

- El sistema AFS almacena en caché los archivos utilizando fragmentos de gran tamaño (64 KB).
- Los otros sistemas analizados en este capítulo permiten almacenar en caché bloques individuales, a medida que sean demandados por los clientes.
- Incrementar el tamaño de la unidad de caché permite aumentar la tasa de aciertos, pero también incrementa la penalización correspondiente a los fallos, porque cada fallo requerirá transferir más datos.
- También incrementa la posibilidad de que aparezcan problemas de coherencia.

Ubicación de la caché

¿ Dónde deben almacenarse los datos de caché, en el disco o en la memoria principal ?

- Las cachés de disco tienen una clara ventaja sobre las cachés de memoria principal: son bastante más fiables.
- Las modificaciones efectuadas en los datos almacenados en caché se perderán cuando se sufra un fallo catastrófico de la máquina, si la caché se almacena en memoria volátil.
- Además, si los datos de caché se almacenan en disco, seguirán estando allí durante la recuperación, y no será necesario volver a extraerlos.
- Las cachés de memoria principal tienen, de todos modos, varias ventajas:

Ubicación de la caché

Las cachés de memoria principales permiten que las estaciones de trabajo no utilicen disco.

- Resulta más rápido acceder a los datos almacenados en una caché de memoria principal que a los de una caché de disco.
- La tecnología está evolucionando para dar memorias de mayor tamaño y menor coste.
- Las previsiones son que las ventajas de velocidad que se podrán conseguir de esta manera compensarán con creces las ventajas de la caché de disco.

Ubicación de la caché

Las cachés de servidor (utilizadas para acelerar la E/S de disco) residirán en memoria principal, independientemente de dónde residan las cachés de usuario; si se utiliza también cachés de memoria principal en las máquinas de usuario, se puede definir un único mecanismo de cache para utilizarlo tanto en los servidores como en los clientes.

Política de actualización de la caché

La política que se utilice para escribir los bloques de datos modificados en la copia maestra del servidor tiene un efecto crítico sobre la fiabilidad y las prestaciones del sistema.

- La política más simple consiste en escribir los datos en disco en cuanto se los coloca en cualquier caché.
- La ventaja de esta política de escritura directa es la fiabilidad: es muy poca la información que se pierde cuando un sistema cliente sufre un fallo catastrófico.

Política de actualización de la caché

Sin embargo, esta política requiere que cada acceso de escritura espere hasta que se envíe la información al servidor, por lo que la velocidad de escritura es muy baja.

- El almacenamiento en caché con un mecanismo de escritura directa es equivalente a utilizar un servicio remoto para los accesos de escritura y aprovechar la caché únicamente para los accesos de lectura.

Política de actualización de la caché

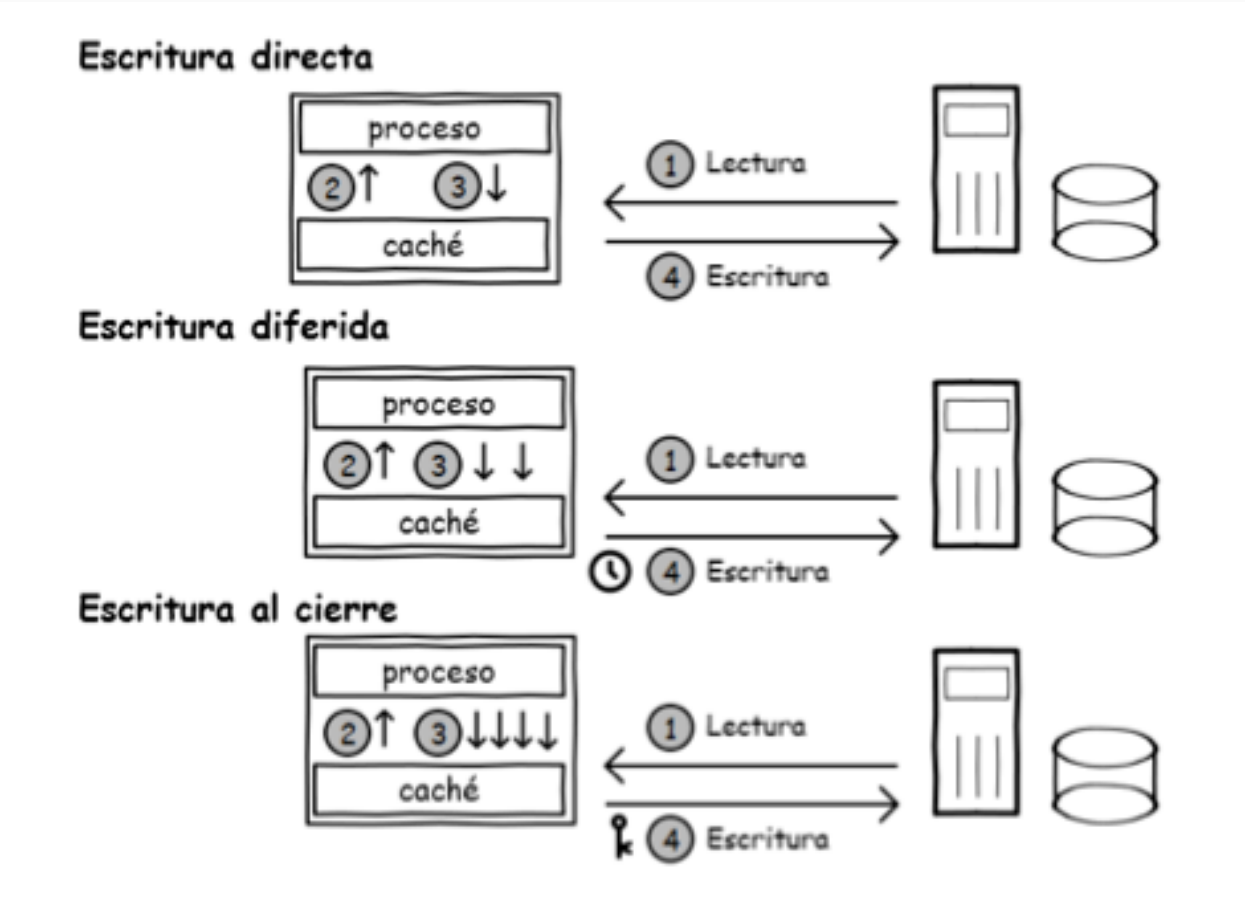

Figure 3:

Política de actualización de la caché

Una alternativa es la política de escritura diferida, también denominada *caché de escritura diferida*; con este tipo de política, lo que se hace es retardar las actualizaciones de la copia maestra.

- Las modificaciones se escriben en la caché y luego se envían al servidor en un momento posterior.
- Esta política tiene dos ventajas sobre la de escritura directa.
- En primer lugar, puesto que las, escrituras se realizan en la caché, los accesos de escritura se completan mucho más rápidamente. En segundo lugar, los datos pueden ser sobrescritos antes de enviarlos al servidor, en cuyo caso sólo será necesario escribir en el servidor la última actualización.
- Lamentablemente, los esquemas de escritura diferida introducen problemas de fiabilidad, ya que los datos no escritos se perderán si una máquina de usuario se cae.

Política de actualización de la caché

Las diversas variantes de la política de escritura diferida se diferencian en lo que respecta al momento en que se vuelcan en el servidor los bloques de datos modificados.

- Una alternativa consiste en volcar un bloque cuando esté a punto de ser expulsado de la caché de cliente.
- Esta opción puede proporcionar unas buenas prestaciones, pero algunos bloques pueden permanecer en la caché de cliente durante mucho tiempo antes de escribirlos en el servidor.
- Un compromiso entre esta alternativa y la política de escritura directa consiste en explorar la caché a intervalos regulares y volcar aquellos bloques que hayan sido modificados desde la última exploración; utilizando el mismo método que UNIX emplea para explorar su caché local.

Política de actualización de la caché

El sistema *Sprite* utiliza esta política, con un intervalo de 30 segundos.

- El sistema *NFS* emplea también esta política para los datos de archivo, pero una vez que se realiza una escritura en el servidor durante un volcado de caché, es necesario esperar a que esa escritura se realice de forma efectiva en el disco del servidor antes de poder considerarla completa.
- NFS trata los metadatos (datos de directorio y datos de atributos de los archivos) de forma diferente.
- Los cambios en los metadatos se envían de manera síncrona al servidor. De este modo, se evita la pérdida de información sobre la estructura de archivos y la corrupción de la estructura de directorios cuando se produce un fallo catastrófico en el cliente o en el servidor.

Política de actualización de la caché

Para NFS con *cachefs*, las escrituras también se realizan en caché de disco local en el momento de llevarlas a cabo en el servidor, con el fin de mantener la coherencia entre todas las copias.

- Por tanto, NFS con cachefs mejora las prestaciones con respecto al sistema NFS estándar en las solicitudes de lectura para las que haya un acierto de caché cachefs pero reduce las prestaciones para las solicitudes de lectura o de escritura en las que se produzca un fallo de caché.

Política de actualización de la caché

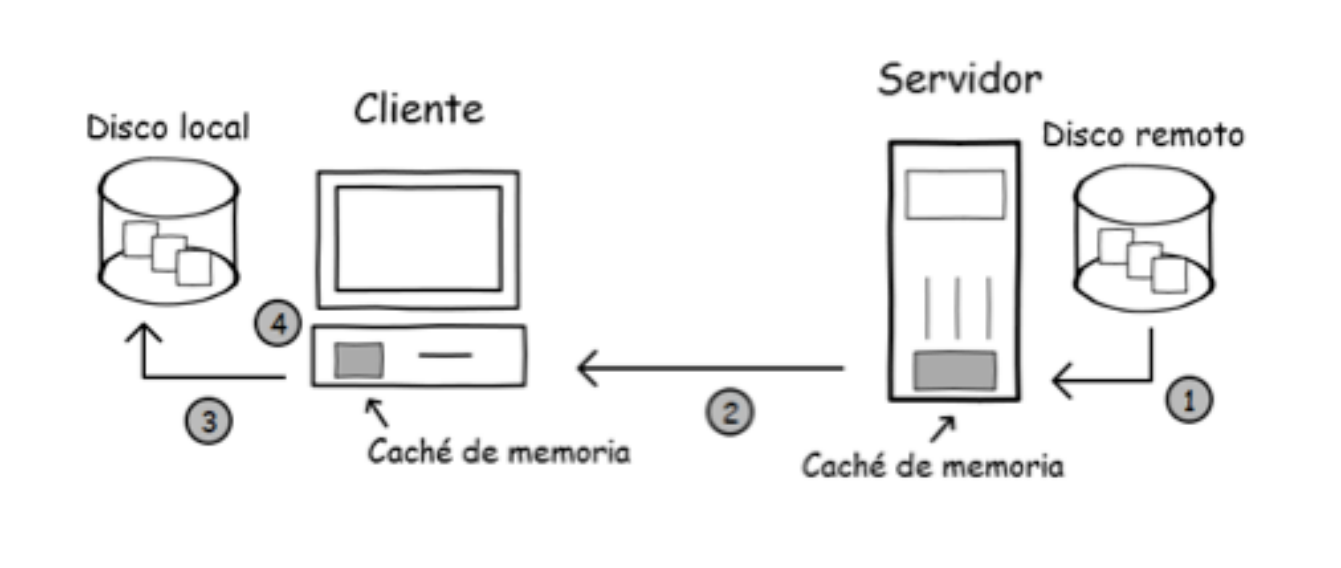

Figure 4:

Política de actualización de la caché

Otra variante de la escritura diferida consiste en escribir los datos en el servidor en el momento de cerrar el archivo.

- Esta política de *escritura durante el cierre* se utiliza en el sistema *AFS*.
- En el caso de los archivos que se abran durante cortos períodos de tiempo o que se modifiquen muy raramente, esta política no reduce significativamente el tráfico de red.

Política de actualización de la caché

Además, la política de escritura durante el cierre requiere que el proceso que realiza el cierre se quede a la espera mientras que se está escribiendo el archivo, lo que reduce la ventaja de prestaciones ofrecida por las escrituras diferidas.

- Sin embargo, para los archivos que están abiertos durante largos períodos de tiempo o que se modifiquen frecuentemente, las ventajas de velocidad de esta política con respecto a la escritura diferida con un volcado más frecuente resultan bastante evidentes.

Coherencia

Las máquinas cliente se enfrentan al problema de decidir si una copia de los datos almacenada en la cache local es coherente o no con la copia maestra (y puede por tanto ser utilizada).

- Si la máquina cliente determina que los datos de su cache están desactualizados, no podrá ya darse servicio a las solicitudes de acceso utilizando dichos datos de la cache.
- Será necesario almacenar en la caché una copia actualizada de los datos.

Coherencia

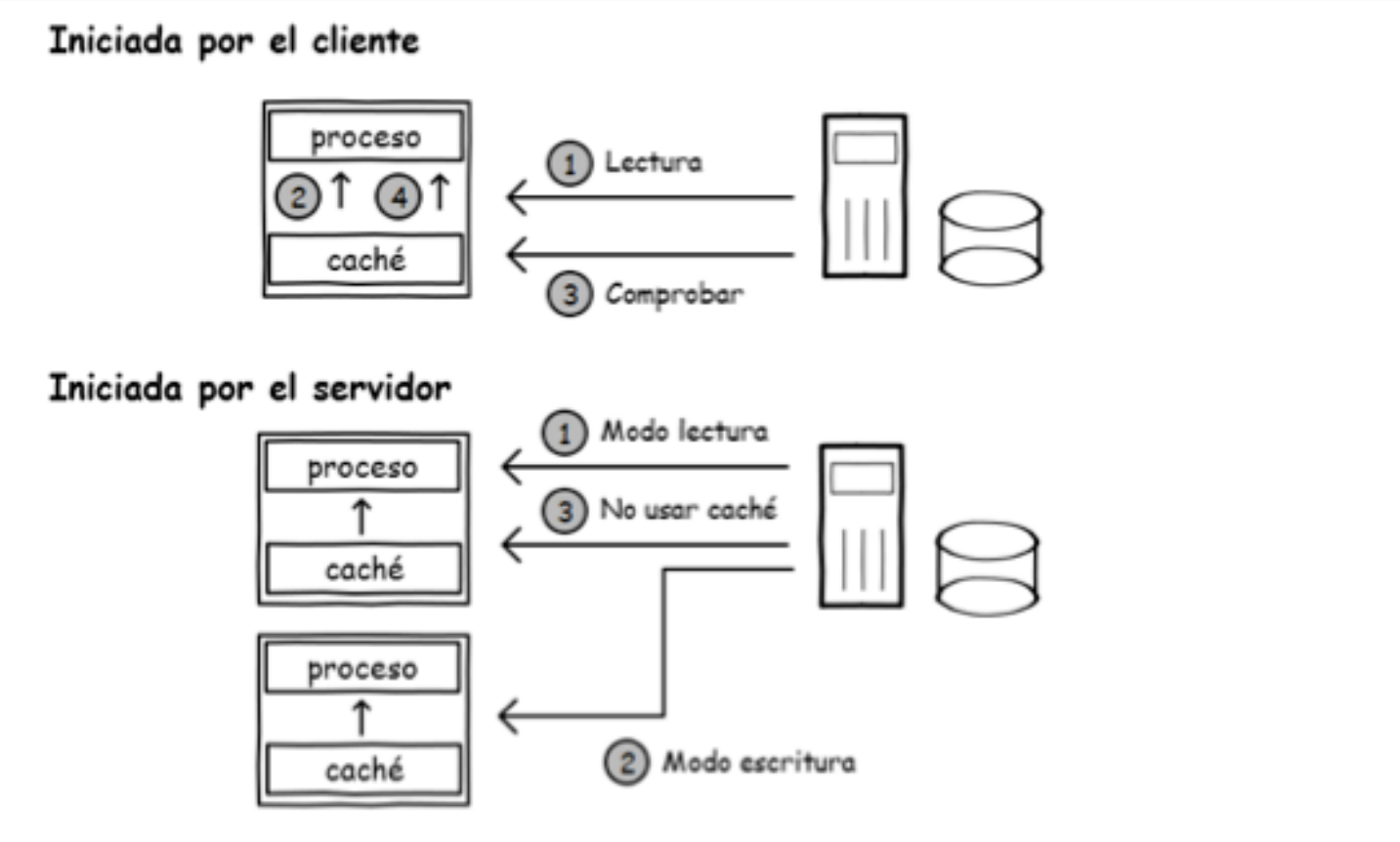

Figure 5:

Coherencia

Existen dos técnicas para verificarla validez de los datos almacenados en la caché.

- Iniciado por parte cliente
- Iniciado por parte del servidor

Iniciado por parte cliente

El cliente inicia una comprobación de validez en la que contacta con el servidor y comprueba si los datos locales son coherentes con la copia maestra.

- La frecuencia de esta comprobación de validez es el aspecto fundamental de esta técnica y determina la semántica de coherencia resultante.
- Esa frecuencia puede variar, pudiendo realizarse la comprobación para cada acceso o solo en el primer acceso a un archivo (básicamente durante la apertura de un archivo); también puede utilizarse cualquier otra frecuencia entre estos dos extremos.

Iniciado por parte cliente

Todos los accesos que lleven aparejada una comprobación de validez sufrirán un cierto retardo, comparados con los accesos a los que se de servicio inmediatamente utilizando los datos de la caché.

- Alternativamente, esas comprobaciones pueden iniciarse a intervalos de tiempo fijos.
- Dependiendo de su frecuencia, las comprobaciones de validez pueden imponer una gran carga tanto a la red como al servidor.

Iniciado por parte del servidor

El servidor registra, para cada cliente, los archivos (o partes de los mismos) que estos tienen almacenados en cache.

- Cuando el servidor detecta una incoherencia potencial, debe reaccionar a la misma.
- Una posibilidad de incoherencia se produce cuando dos clientes distintos que operan con modos conflictivos almacenan en caché un mismo archivo.
- Si se implementa la semántica de UNIX, se puede resolver la incoherencia potencial haciendo que el servidor juegue un papel activo.

Iniciado por parte del servidor

El servidor deberá ser notificado cada vez que se abra un archivo, y deberá indicarse el modo deseado (lectura o escritura) para cada apertura archivo.

- El servidor puede entonces actuar cuando detecte que el archivo ha sido abierto simultáneamente en modos conflictivos, desactivando en ese caso el mecanismo de caché para ese archivo concreto.
- En la práctica, la desactivación del mecanismo de caché provoca la conmutación a un modo de operación basado en el paradigma de servicio remoto.

Servicios con y sin memoria del estado

Existen dos técnicas para almacenar la información del lado del servidor cuando un cliente accede a archivos remotos.

- O bien el servidor lleva la cuenta de cada archivo al que esté accediendo cada cliente, o simplemente se limita a proporcionar bloques a medida que los clientes los solicitan, sin ningún tipo de conocimiento de cómo se utilizan esos bloques.
- En el primero de los casos, el servicio proporcionado tiene memoria del estado, en el segundo caso, no tiene memoria del estado.

Servicios con y sin memoria del estado

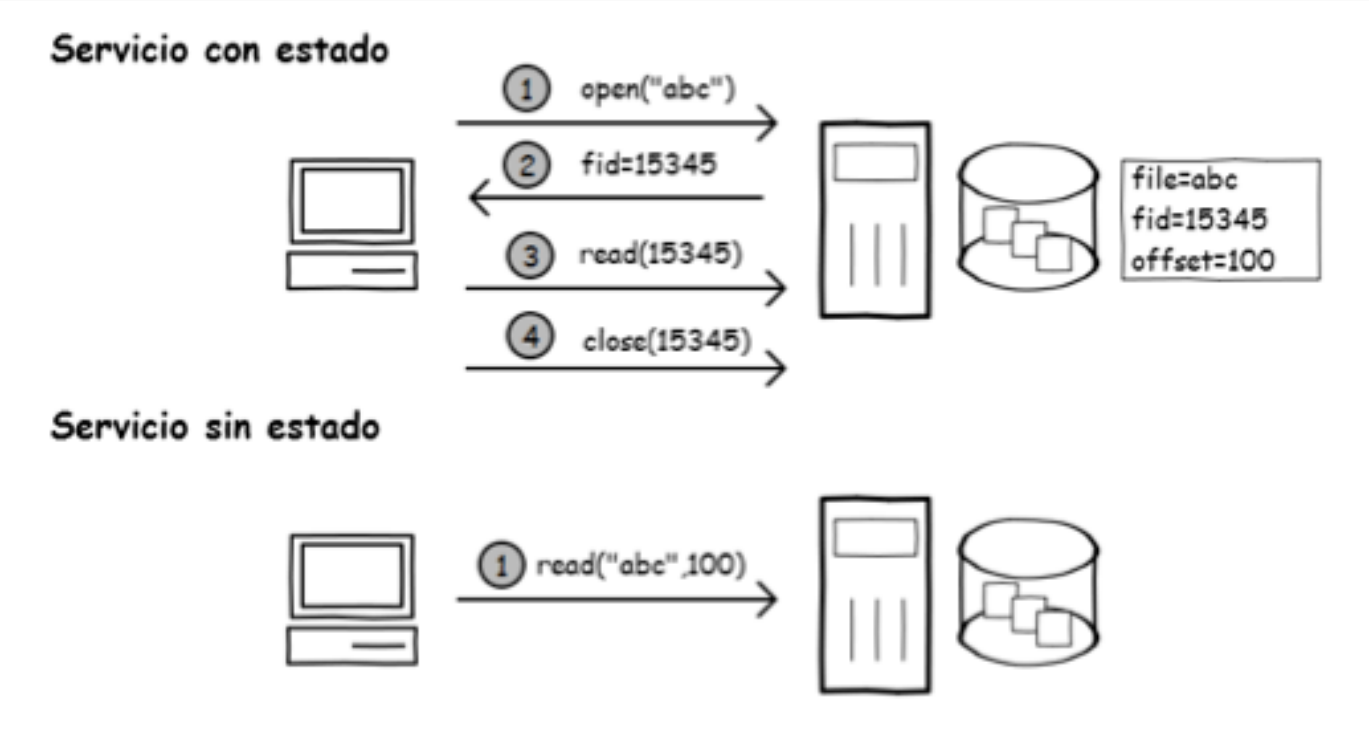

Figure 6:

Servicios con y sin memoria del estado

El escenario típico de un servicio de archivos con memoria del estado sería el siguiente: un cliente debe realizar una operación *open()* sobre un archivo antes de acceder a dicho archivo.

- El servidor extrae la información acerca del archivo desde su disco, la almacena en la memoria y proporciona al cliente un identificador de conexión que es exclusivo de ese cliente y de ese archivo concreto que se ha abierto.
- En términos UNIX, el servidor extrae el inodo y proporciona al cliente un descriptor de archivo que sirve como índice para una tabla interna de inodos.
- Este identificador se emplea para los accesos subsiguientes hasta que termina la sesión.
- Un servicio con memoria del estado está caracterizado como una conexión entre el cliente y el servidor durante la sesión.

Servicios con y sin memoria del estado

Durante el cierre del archivo, o mediante un mecanismo de recolección de memoria, el servidor deberá en un momento u otro reclamar el espacio de memoria principal utilizado por los clientes que ya no estén activos.

- El punto clave en lo que respecta a la tolerancia a fallos en un servicio con memoria del estado es que el servidor mantiene información en memoria principal acerca de sus clientes.
- El sistema *AFS* es un servicio de archivos con memoria del estado.

Servicios con y sin memoria del estado

Un servicio de archivos sin memoria del estado evita la información de estado haciendo que cada solicitud sea autocontenida.

- En otras palabras, cada solicitud identifica el archivo y la posición dentro del archivo (para los accesos de lectura y escritura) de modo completo.
- El servidor no necesita mantener una tabla de archivos abiertos en memoria principal, aunque usualmente lo hace por razones de eficiencia. Además, no hay necesidad de establecer y terminar una conexión mediante operaciones *open()* y *close()*.

Servicios con y sin memoria del estado

Estas operaciones son totalmente redundantes, ya que cada operación de archivo es totalmente autónoma y no se considera parte de una sesión.

- El proceso cliente abriría el archivo y dicha apertura no provocaría el envío de mensajes remotos.
- Las lecturas y escrituras tendrían lugar como mensajes remotos (o búsquedas en caché).
- El cierre final por parte del cliente implicaría únicamente, de nuevo, una operación local. *NFS* es un servicio de archivos sin memoria del estado

Servicios con y sin memoria del estado

La ventaja de un servicio con memoria del estado frente a otro que no lo tenga es el incremento en las prestaciones.

- La información acerca de los archivos se almacena en caché dentro de la memoria principal, con lo que se puede acceder fácilmente a la misma utilizando el identificador de conexión, evitándose así accesos a disco.
- Además, un servidor con memoria del estado sabe si un archivo está abierto para acceso secuencial y puede, por tanto, leer de manera anticipada los bloques siguientes.
- Los servidores sin memoria del estado no pueden hacer esto, ya que no tienen ningún conocimiento de cuál es el propósito de las solicitudes de los clientes.

Servicios con y sin memoria del estado

La distinción entre un servicio con memoria del estado y otro que no lo tenga resulta más evidente cuando se considera los efectos de un fallo catastrófico que tenga lugar durante la actividad de servicio.

- Un servidor con memoria del estado perderá todos sus datos de estado volátiles durante el fallo.
- Para poder efectuar una recuperación grácil del servidor será necesario restaurar su estado, usualmente mediante un protocolo de recuperación basado en un diálogo con los clientes.

Servicios con y sin memoria del estado

Otras formas de recuperación menos gráciles requieren que se aborten las operaciones que estuvieran teniendo lugar en el momento de producirse el fallo.

- En el caso de que lo que fallen sean los clientes, el problema es distinto.
- El servidor necesitará darse cuenta de que se ha producido ese fallo para poder reclamar el espacio asignado para registrar el estado de los procesos del cliente que ha fallado. Este fenómeno se denomina en ocasiones detección y eliminación de huérfanos.

Servicios con y sin memoria del estado

Los servidores sin memoria del estado no presentan estos problemas, ya que el servidor puede, después del arranque, responder a cualquier solicitud autocontenida sin ninguna dificultad.

- Por tanto, los efectos de los fallos de servidor y de los mecanismos correspondientes de recuperación son prácticamente inapreciables.
- No existe ninguna diferencia entre un servidor lento y un servidor que esté efectuando una recuperación desde el punto de vista del cliente.
- El cliente continuará retransmitiendo su solicitud en caso de no recibir ninguna respuesta.

Servicios con y sin memoria del estado

La desventaja de utilizar el robusto servicio de una máquina sin memoria del estado es que los mensajes de solicitud son mas largos y el procesamiento de las solicitudes mas lento, ya que no se dispone de ninguna información interna para acelerar el procesamiento.

- Además, el servicio sin memoria del estado impone restricciones adicionales al diseño del DFS.
- En primer lugar, puesto que cada de solicitud identifica el archivo de destino, es necesario utilizar un esquema de nombrado de bajo nivel que sea global y uniforme para todo el sistema.
- Traducir los nombres remotos en nombres locales para cada solicitud hace que el procesamiento de las solicitudes sea todavía más lento.

Servicios con y sin memoria del estado

En segundo lugar, puesto que los clientes retransmiten las solicitudes relativas a operaciones de archivos, estas operaciones deben ser idempotentes; es decir, cada operación debe tener el mismo efecto y devolver la misma salida si se ejecuta varias veces consecutivamente.

- Los accesos de lectura y escritura autocontenidos son idempotentes, siempre y cuando utilicen un contador absoluto de bytes para indicar la posición dentro del archivo a la que quieren acceder, y que no dependan de un desplazamiento incremental (como se hace en las llamadas al sistema *read()* y *write()* de UNIX).
- Sin embargo, se debe tener cuidado a la hora de implementar las operaciones destructivas (como por ejemplo el borrado de un archivo) con el fin de hacerlas también idempotentes.

Replicación de archivos

La replicación de archivos en diferentes máquinas dentro de un sistema de archivos distribuido constituye un útil mecanismo de redundancia para mejorar la disponibilidad.

- La replicación multimáquina puede aumentar también las prestaciones: seleccionar una réplica cercana para dar servicio a una solicitud de acceso da como resultado un tiempo de servicio más corto.

Replicación de archivos

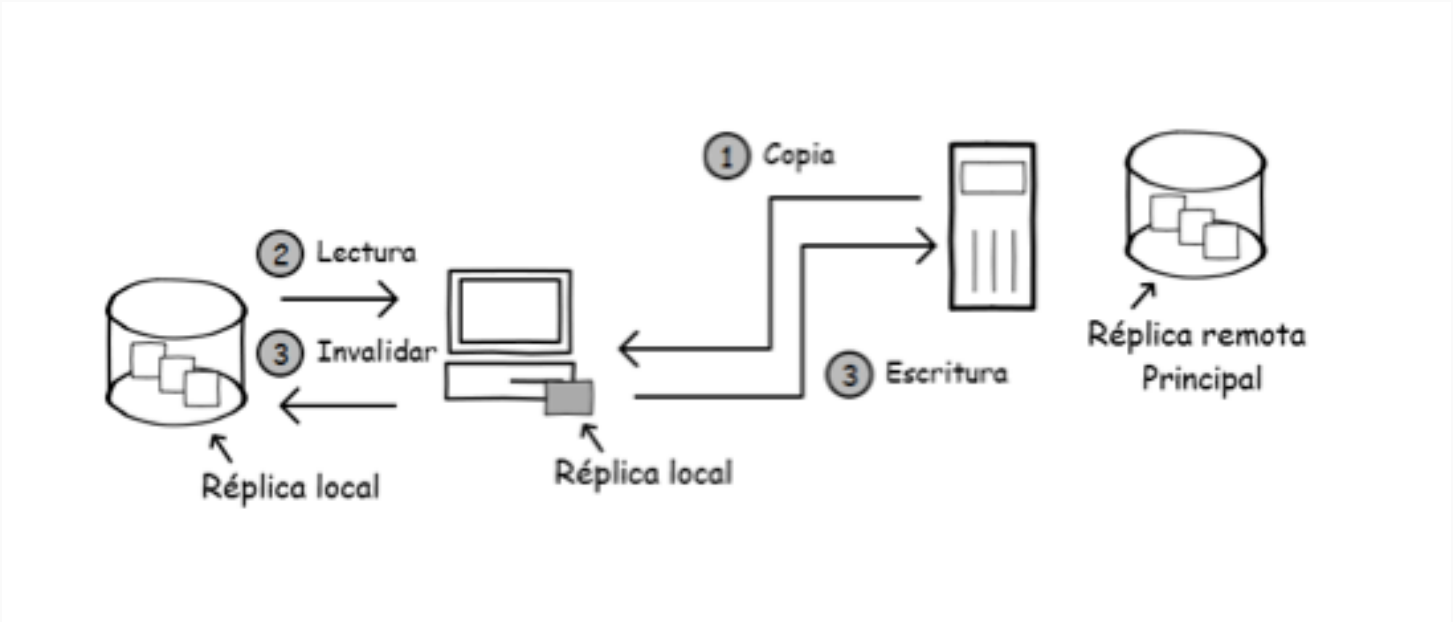

Figure 7:

Replicación de archivos

El requisito básico de un esquema de replicación es que las diferentes réplicas del mismo archivo residan en máquinas que sean independientes en lo respecta a los fallos, es decir, que la disponibilidad de una réplica no se vea afectada por la disponibilidad del resto de las réplicas.

- Este requisito obvio implica que la gestión de la replicación es, inherentemente, una actividad opaca en lo que respecta a la ubicación.
- Será necesario proporcionar mecanismos para poder colocar una réplica en una máquina concreta.

Replicación de archivos

Resulta deseable ocultar los detalles de la replicación a ojos de los usuarios.

- El establecimiento de la correspondencia entre un nombre de archivo replicado y una réplica concreta es tarea del esquema de nombrado.
- La existencia de réplicas debe ser invisible para los niveles superiores.
- Sin embargo, en los niveles inferiores, es necesario distinguir unas réplicas de otras utilizando nombres de bajo nivel distintos.

Replicación de archivos

- Otro requisito de transparencia es que se debe proporcionar mecanismos de control de la replicación en los niveles superiores.
- Esos mecanismos de control de la replicación incluyen poder determinar el grado de replicación y la colocación de las réplicas.
- En determinadas circunstancias, puede que convenga proporcionar estos detalles a los usuarios.

Replicación de archivos

El problema principal asociado con las réplicas es su actualización.

- Desde el punto de vista de un usuario, todas las réplicas de un archivo denotan la misma entidad lógica, por lo que cualquier actualización en una réplica debe reflejarse en todas las réplicas restantes.
- De manera más precisa, será necesario preservar la semántica de coherencia relevante cuando los accesos a las réplicas se contemplen como accesos virtuales a los archivos lógicos de las réplicas.

Replicación bajo demanda

Algunos sistemas soportan un mecanismo de replicación bajo demanda, que es una política de control de replicación automático similar a los mecanismos de almacenamiento en caché de archivos completos.

- Con una replicación bajo demanda, la lectura de una réplica no local, hace que esa réplica se almacene localmente en la caché, generando así una nueva réplica no principal.
- Las actualizaciones se realizan únicamente en la réplica principal y hacen que las demás réplicas queden invalidadas, enviándose los oportunos mensajes.

Replicación bajo demanda

- No se garantiza la invalidación atómica y serializada de todas las réplicas no principales. Por ello, puede darse el caso de que una réplica obsoleta se considere válida.
- Para satisfacer los accesos de escritura remotos, lo que se hace es migrar la copia principal hasta la máquina solicitante.

Referencia bibliográfica

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap.18, 9a Edición, John Wiley, 2013.