Tema 1:

Características de los Ficheros Distribuidos

Propiedades

- Proporciona almacenamiento de información permanente
- Identifica los ficheros en un espacio de nombres (normalmente estructurado)
- Es posible el acceso concurrente desde varios procesos
- En sistemas multiusuario proporciona protección de accesos.
- Transparencia en la identificación.
  - Espacio de nombres único e independiente del cliente.
- Transparencia en la ubicación.
  - Para permitir la movilidad del fichero de una ubicación a otra, se requiere una correspondencia dinámica nombre-ubicación.
- Escalabilidad.
  - Espacios de nombres estructurados, y replicación (caching) para evitar cuellos de botella.

Propiedades

- Robustez ante fallos.
  - El servidor no debe verse afectado por los fallos de los clientes,
    - `gestión del estado de los clientes en el servidor.`
  - La interfaz ofrecida a los clientes debe proporcionar en lo posible operaciones idempotentes (f(f(x)) = f(x)), que garanticen la corrección ante invocaciones repetidas al servidor.
- Disponibilidad.
  - Un aspecto de la disponibilidad es permitir el funcionamiento en modo desconectado, que requiere caching de ficheros enteros
- Tolerancia a fallos.
  - Implican alguna forma de replicación.
- Consistencia.
  - El objetivo es mantener en lo posible la semántica de los sistemas centralizados,
    - `por ejemplo, preservar la semántica UNIX en presencia de caching u otras formas de replicación.`
- Seguridad.
  - La necesidad de autenticación remota implica nuevos modelos de protección, basados en credenciales en lugar de listas de accesos.

Caracterización del uso de los ficheros

- El diseño de un sistema de ficheros distribuido que proporcione un buen nivel de rendimiento deberá basarse en las características de uso de los ficheros por las aplicaciones
  - La mayoría de los ficheros son de pequeño tamaño.
    - Esto implica que 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.
      - De hecho ni siquiera la semántica UNIX en sistemas centralizados gestiona la sincronización entre varios escritores, pasando la responsabilidad a las aplicaciones.
    - Puede ser rentable una gestión optimista del caching y la replicación, que presuponga que hay un único escritor y rectifique en caso de detectar a posteriori escrituras simultáneas.

Caracterización del uso de los ficheros

- El ratio búsqueda/uso suele ser bajo.
  - En las bases de datos este ratio es elevado porque las aplicaciones suelen acceder a una gran cantidad de elementos y en la mayoría de los casos para una simple consulta
  - En ficheros es habitual realizar un proceso más o menos costoso para cada elemento accedido
    - Este hecho favorece el caching
- 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

Modelo

- El modelo de sistema de ficheros distribuido que vamos a definir aquí considera una estructura cliente-servidor que incluye:
  - Servicios de nombres
  - Servicios de ficheros
  - Un mecanismo de identificación única de los ficheros por los clientes.

- Estructura
  - Cliente. Es la interfaz local con la aplicación.
    - Interpreta las llamadas al sistema sobre ficheros
    - 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).

Modelo

- Estructura
  - Servicio de ficheros
    - Mantiene el contenido de los ficheros (y directorios) y los atributos de los ficheros:
      - tiempos de creación
      - tiempo de último acceso
      - última modificación
      - longitud
      - cuenta de referencias
    - Un fichero se identifica en el servicio de Ficheros mediante un identificador único de fichero, UFID.
    - Las operaciones sobre un fichero se refieren explícitamente a su UFID.
    - La interfaz del servicio de ficheros con el cliente ofrece operaciones como
      - *leer, escribir, crear, borrar,*
      - *obtener_atributos y modificar_atributos.*

Modelo

- Estructura
  - Servicio de nombres (directorios)
    - Es el encargado de proporcionar transparencia en la ubicación.
    - En general, es una base de datos con elementos (nombre, UFID), donde se crean, se modifican y se buscan entradas.
    - El nombre viene especificado por el string de caracteres que describe el path.
    - La interfaz del servicio de nombres ofrece al cliente operaciones de
      - buscar_nombre,
      - añadir_nombre,
      - borrar_nombre.
    - Algunos atributos del fichero se mantienen por el servicio de nombres:
      - tipo de fichero (ordinario o directorio)
      - identificador del usuario
      - propietario del fichero
      - derechos de acceso.

Modelo

- Identificación de ficheros
  - El cliente especifica un fichero al servidor de ficheros mediante el identificador único de ficheros, UFID, expedido por el servidor de ficheros cuando se crea un fichero
  - el UFID requiere identificar:
    - El servidor del fichero, S.
    - El fichero dentro del servidor, F.
    - Los derechos de acceso sobre el fichero, D.
  - Los UFIDs deben protegerse de la manipulación por el cliente,
    - El cliente podría generar un UFID con derechos de acceso falsos.
    - En un sistema centralizado tipo UNIX el identificador del proceso que accede al fichero determina directamente el identificador del usuario (UID), almacenado en el inodo del fichero, lo que autentifica al cliente.
    - En sistemas distribuidos se requiere un mecanismo adicional de autentificación que garantice la aplicación segura de los derechos de acceso.

Modelo

- Identificación de ficheros
  - Ejemplo de modelo de identificación similar a Amoeba
    - Cuando se crea un fichero, el servidor de ficheros genera un número aleatorio, R, que se añade al UFID.
    - El UFID expedido por el servidor de nombres para un fichero solicitado por un cliente incluye un campo **C** que es una codificación de R y los derechos de acceso para ese cliente D
    - La codificación se realiza mediante una función unidireccional, fc:

      $$C = fc\ (R,\ D)$$

    - Para comprobar la validez de un acceso a un fichero, el servidor del fichero
      - aplicará la función fc sobre
        - R almacenado en el servidor
        - D del UFID del cliente,
      - y comparará el resultado con el campo C del UFID del cliente:

        $$¿\ fc\ (R,\ D) = C$$

    - Un cliente que reclame unos derechos de acceso distintos a los reconocidos no superará esta comprobación.

Modelo

- Identificación de ficheros
  - Ejemplo de modelo de identificación similar a Amoeba

nombre	UFID		
/dir1/dir2/fich	*S*	*F*	*R*

UFID del fichero `/dir1/dir2/fich`
almacenamiento en el Servidor de Nombres

S	*F*	*C*	*D*

UFID de la petición del cliente

¿ fc (R, D) = C ?

Servidores de nombres

- El servidor de nombres permite al cliente determinar la ubicación de un fichero (UFID) a partir de su PATH
- Estrictamente un es servicio de directorios.
- El árbol de directorios se divide en dominios
- Un ***dominio*** está asociado a una parte del árbol de directorios o PATH
- Un servidor de nombres gestiona uno o más dominios
- El cliente mantiene una tabla (dominio (PATH), servidor de nombres)
  - Dado un PATH el cliente busca en la tabla el servidor de nombres asociado
- Un servidor de nombres puede resolver el PATH como asociado a un dominio mantenido por otro servidos de nombres.
  - Ocurre en redes de área amplia
  - Este encadenamiento de peticiones a diferentes servidores de nombres se conoce como ***navegación***

Servidores de nombres

- Esquemas de navegación
  - Iterativa.
    - El servidor de un dominio resuelve su parte del path y responde al cliente indicando el nuevo servidor para resolver el resto del path.
    - La comunicación se resuelve mediante RPC convencional.
    - Esta alternativa está limitada a un dominio administrativo único, ya que el cliente puede no tener acceso fuera de él.
    - Un ejemplo es el servidor NIS (Network Information Service), utilizado por NFS.
      - NIS no es un servicio de directorios que se ajuste al modelo definido, ya que no gestiona UFIDs, sino un servicio de nombres más general que resuelve la navegación.
    - Una variante es la navegación *iterativa controlada por el servidor*.
      - En este caso, el servidor invocado por el cliente es el encargado de realizar (iterativamente) la secuencia de peticiones para la resolución del path, respondiendo al cliente cuando éste ha sido resuelto completamente.

Servidores de nombres

- Esquemas de navegación
  - Recursiva.
    - El servidor de un dominio resuelve su parte del PATH y cursa una petición a un nuevo servidor para resolver el resto del PATH.
    - El servidor que resuelve el último elemento del path es el que responde al cliente,
      - por lo que este esquema no admite RPC convencional.
    - Requiere menos comunicación que el esquema anterior.
    - Un ejemplo es el servidor DNS (Domain Name System), que se utiliza en internet
      - también soporta la navegación iterativa
    - Una alternativa con mayor latencia es que el retorno siga el camino inverso nodo a nodo
      - tiene la ventaja de que los nodos pueden hacer caching de los nombres resueltos.

Servidores de nombres

- Esquemas de navegación
  - Para disminuir latencias, los servicios de nombres hacen un extenso uso del caching,
    - lo que conduce a situaciones de inconsistencia entre los servidores de nombres.
  - Como la migración de dominios entre servidores es infrecuente, no suele ser un objetivo prioritario del servicio de nombres el prevenir las posibles situaciones de identificación errónea de dominios debidas al caching.

Servidores de ficheros

- 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 sin incurrir en una fuerte penalización en el rendimiento (fundamentalmente la latencia).
- Utilizan extensamente el caching
- Utilizan mecanismos de gestión de las copias que permiten compromisos razonables entre semántica y rendimiento.
- Semanticas de compartición
  - Semántica UNIX.
    - Igual a la que caracteriza el acceso a los ficheros de los sistemas UNIX clásicos.
      - Las operaciones sobre un fichero se ordenan totalmente en el tiempo, lo que implica que una lectura devuelve la última actualización del fichero.
      - Es complejo para un sistema distribuido proporcionar semántica UNIX.

Servidores de ficheros

- Semanticas de compartición
  - Semántica de sesión.
    - 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 el estado del fichero
        - por ejemplo el apuntador a la posición actual
    - En un sistema distribuido la semántica de sesión equivale a trabajar sobre una copia local
      - se carga cuando el fichero se abre y
      - se actualiza en el servidor cuando se cierra.
    - Los conflictos los debe solucionar el usuario o a la aplicación.

Servidores de ficheros

- Semanticas de compartición
  - Ficheros inmutables.
    - Los ficheros no se modifican, sino que se reemplazan atómicamente por nuevas versiones
    - los directorios sí se modifican.
    - Es posible la compartición concurrente para lectura
    - Si se pretende acceder un fichero abierto por otro proceso para escritura existen dos alternativas:
      A) considerar error la operación de abrir,
      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, subversion).

Servidores de ficheros

- Semanticas de compartición
  - Semántica de transacciones.
    - Una transacción esta formada por una serie de operaciones entre las instrucciones BEGIN TRANSACTION y END TRANSACTION
    - Tiene carácter "atómico" (se realiza completo o nada)
    - El uso de transacciones permite definir explícitamente secuencias de operaciones sobre ficheros
    - Las propiedades ACID de las transacciones garantizan las operaciones críticas sobre ficheros
    - el acrónimo ACID responde a las siguientes propiedades
      - atomicidad (atomicity),
      - coherencia (consistency)
        - transformando un estado coherente de datos en otro estado de datos igualmente coherente
      - aislamiento (isolation),
        - Se comportan como si el sistema ejecutara de forma única cada transacción
      - permanencia (durability).
        - Una transacción es una unidad de recuperación.
        - Si una transacción se realiza satisfactoriamente, el sistema garantiza que sus actualizaciones se mantienen aunque el equipo falle inmediatamente después de la confirmación.

Servidores de ficheros

- Tipos de servidores
  - 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
    - Servidor sin estados.
      - El servidor no almacena información del cliente.
      - El cliente suministra en cada llamada toda la información necesaria para realizar la operación
        - incluido el puntero a la posición de acceso actual.
      - Ventajas:
        - Un fallo del cliente no afecta al servidor
      - Inconvenientes:
        - Hace difícil proporcionar una semántica UNIX
          - por ejemplo, los derechos de acceso al fichero se comprueban en cada acceso
      - El ejemplo más notable de servidor sin estados es NFS

Servidores de ficheros

- Tipos de servidores
  - Servidor con estados.
    - El servidor crea una entrada para el fichero ante una invocación de abrir fichero de un cliente.
    - El servidor almacena
      - el apuntador a la última posición accedida y
      - otra información,
      - hasta bloques del fichero,
        - lo que posibilita lectura anticipada en el servidor.
    - Ventajas:
      - Las sucesivas invocaciones de acceso al fichero requieren mensajes más cortos
    - Inconvenientes:
      - Limita de forma inherente el número de ficheros abiertos simultáneamente.
      - El servidor tiene que gestionar el posible fallo de los clientes, liberando los recursos
    - Un ejemplo de servidor con estado es RFS (*Remote File System*), actualmente muy poco utilizado.

Servidores de ficheros

- Tipos de servidores
  - Las operaciones en sistemas de ficheros sin estado, tienden a ser idempotentes *f(f(x))=f(x)*
    - Esto permite una gestión más sencilla de las reinvocaciones ante sospechas de fallo en la transmisión de la petición
  - Por ejemplo, una operación de lectura
    - para un servidor con estado

      status= leer_fich (ufid, buffer, longitud)

    - para un servidor sin estado

      status= leer_fich (ufid, buffer, posición, longitud)

  - Si, tras expirar el time-out de espera de una respuesta, el cliente reenvía la invocación con los mismos parámetros, el comportamiento difiere en ambos tipos de servidores
    - En un servidor sin estado se indica la posición y la operación tendrá idéntico resultado a la fallida
    - En un servidor con estado, si en el servidor se ha modificado la longitud con la primera llamada, la segunda no tendrá el resultado esperado

Caching y gestión de la consistencia

- Almacenamiento temporal de (trozos de) ficheros en el nodo cliente, con el objetivo de minimizar los costes de comunicación
  - El servidor puede proporcionar también *buffer cache* de bloques como en cualquier sistema de ficheros centralizado
- La unidad de gestión o cantidad de información que se transmite en cada petición, puede ser
  - el fichero completo
  - un bloque del fichero
    - El tamaño de éste no tiene que coincidir con el tamaño del bloque del sistema local de ficheros
- Transmitir cantidades grandes de información proporciona **lectura anticipada** (*buffering*), que potencia también la localidad espacial.
- Tipos según el almacenamiento
  - Persistente
    - En disco
  - No persistente
    - En memoria
      - Memoria del núcleo
      - Memoria de usuario

Caching y gestión de la consistencia

- Caching estructurado
  - Es una generalización del almacenamiento temporal, en redes WAN,
  - Consiste en almacenamiento temporal en nodos intermedios a diferentes niveles.
  - Este servicio es el que ofrecen los nodos *proxy* en Internet.
- El caching, como toda forma de replicación, lleva asociada la necesidad de gestionar la consistencia.
- La política de gestión de la consistencia condiciona la semántica de compartición.
- Políticas básicas para gestionar la consistencia
  - Write-through.
    - Cuando un elemento se modifica, se copia en el servidor.
    - Mejoras
      - escritura retardada (acumulando modificaciones),
      - no copiar en el servidor los ficheros temporales.

Caching y gestión de la consistencia

- Políticas básicas para gestionar la consistencia
  - Write-on-close.
    - El fichero se escribe en el servidor cuando se cierra.
    - Determina semántica de sesión.
    - Mejoras
      - retardar la escritura (es frecuente que un fichero se borre después de cerrar).
  - Gestión centralizada.
    - El servidor de ficheros gestiona la apertura/cierre de ficheros mediante un algoritmo de sincronización lectores-escritores.
    - Proporciona semántica UNIX, pero es poco escalable y no tolerante a fallos.

Caching y gestión de la consistencia

- Políticas básicas de validación
  - Permiten a un nodo conocer cuando la réplica de un fichero que está se usando en su cache ha quedado obsoleta por la actualización de otra réplica del mismo fichero en otro nodo (no necesariamente el servidor)
  - No necesarias en gestión centralizada
  - Dos políticas básicas en función de donde parta la iniciativa
    - Desde los clientes,
      - Accede 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.
    - Desde el servidor,
      - notificando a los clientes cuando una copia ha quedado obsoleta (callback).
      - Requiere almacenar algo de estado en el servidor.

- Propiedades
- Propiedades
- Caracterización del uso de los ficheros
- Caracterización del uso de los ficheros
- Modelo
- Modelo
- Modelo
- Modelo
- Modelo
- Modelo
- Servidores de nombres
- Servidores de nombres
- Servidores de nombres
- Servidores de nombres
- Servidores de ficheros
- Servidores de ficheros
- Servidores de ficheros
- Servidores de ficheros
- Servidores de ficheros
- Servidores de ficheros
- Servidores de ficheros
- Caching y gestión de la consistencia
- Caching y gestión de la consistencia
- Caching y gestión de la consistencia
- Caching y gestión de la consistencia