mas Distribuidos

4 Sistemas de ficheros distribuidos

Índice

- Introducción
- Estructura de un SFD
- Resolución de nombres
- Acceso a los datos
- Gestión de cache
- Gestión de cerrojos
- Estudio de ejemplos: NFS, AFS y Coda
- Sistemas de ficheros paralelos
  - *General Parallel File System* (GPFS)
  - *Google File System* (GFS)

onceptos básicos

eros distribuido (SFD)
heros para sistema distribuido
tos dispositivos en diferentes nodos ofreciendo a
sma visión que un SF centralizado
suarios compartan información de forma transparente
lesde cualquier máquina
ectos similares a SF centralizados
os específicos como por ejemplo:
nombres involucra a varios nodos
afecta a múltiples nodos
lerancia a fallos

Características deseables del SFD

- Aplicables las correspondientes al SD global:
  - Transparencia, fiabilidad, rendimiento, escalabilidad, seguridad.
- Específicamente:
  - Espacio de nombres único
  - Soporte de migración de ficheros
  - Soporte para replicación
  - Capacidad para operar en sistemas "desconectados"
    - Una red "partida" o un cliente que usa un sistema portátil
  - Soporte de heterogeneidad en hardware y S.O.
  - Integración de nuevos esquemas de almacenamiento (SAN)
  - Paralelismo en acceso a datos de un fichero

structura del SFD

on aplicación) – Servidor (nodo con disco)
funcionalidad de SF entre cliente y servidor?
adicional"
rciona acceso a ficheros almacenados en sus discos
la entre aplicación y servidor
enos funcionalidad (clientes *fat* o *thin*)
ternativa"
rciona acceso a bloques de disco
funcionalidad del SF
stemas de ficheros para *clusters*
más adelante

Arquitectura del SFD

- Solución basada en arquitectura tradicional parece sencilla:
  - SF convencional en servidor
  - Exporta servicios locales para abrir, cerrar, leer, escribir, cerrojos,...
- ¿Asunto zanjado? No todo está resuelto:
  - Resolución de nombre de fichero:
    - Cliente y varios servidores involucrados. ¿Cómo se reparten trabajo?
  - Acceso a los datos:
    - ¿se transfiere sólo lo pedido? ¿más cantidad? ¿todo el fichero?
    - Uso de cache en el cliente. Coherencia entre múltiples caches.
  - Gestión de cerrojos:
    - ¿Qué hacer si se cae un cliente en posesión de un cerrojo?
  - Otros: migración, replicación, heterogeneidad, ...

iones sobre los ficheros

hero: Traducción del nombre
mbres
ras sobre el fichero
s
de cerrojos sobre el fichero

Gestión de nombres

- Similar a SF convencionales:
  - Espacio de nombres jerárquico basado en directorios
  - Esquema de nombres con dos niveles:
    - Nombres de usuario (*pathname*) y Nombres internos
  - Directorio: Relaciona nombres de usuario con internos
- Nombres de usuario
  - Deben proporcionar transparencia de la posición
    - Nombre no debe incluir identificación del nodo donde está
      triqui.fi.upm.es:/home/fichero.txt
- Nombres internos
  - Identificador único de fichero (UFID) utilizado por el sistema
    - Puede ser una extensión del usado en SF convencionales.
    - Por ejemplo:
      id. de máquina + id. disco + id. partición + id. inodo

Espacio de nombres

nvencionales:
mbres dividido en volúmenes (o particiones, o ...)
en gestionado por un servidor
mediante composición de volúmenes:
istribuida de operación de montaje de UNIX

la composición:
a de ficheros remoto sobre la jerarquía local (NFS)
cliente: información de montaje se almacena en cliente
nombres diferente en cada máquina
de nombres en todas las máquinas (AFS)
servidor: información de montaje se almacena en servidor
nombres común para el SD

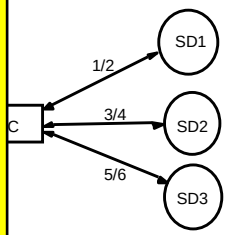
Fernando Pérez Costoya

Resolución de nombres

- Traducir una ruta que se extiende por varios servidores
- ¿Quién busca cada componente de la ruta?
  - cliente: solicita contenido del directorio al servidor y busca (AFS)
  - servidor: realiza parte de la búsqueda que le concierne
- Alternativas en la resolución dirigida por los servidores
  - iterativa, transitiva y recursiva
- "Cache de nombres" en clientes
  - Almacén de relaciones entre rutas y nombres internos
    - También existe en SF convencional
  - Evita repetir proceso de resolución
    - Operación más rápida y menor consumo de red (*escalabilidad*)
  - Necesidad de coherencia
    - Fichero borrado y nombre interno reutilizado
    - Uso de contador de versión del inodo
      id. de máquina + id. disco + id. partición + id. inodo + nº versión

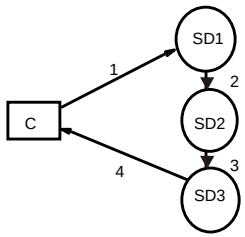
Sistemas Distribuidos 10 — Fernando Pérez Costoya

esolución iterativa



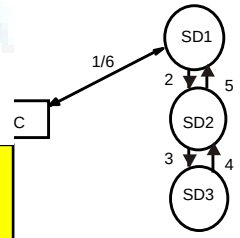
Fernando Pérez Costoya

Resolución transitiva



Sistemas Distribuidos 12 — Fernando Pérez Costoya

solución recursiva



so a datos del fichero

el fichero, se tiene info. para acceder al mismo

seño vinculados con acceso a datos:

- tiza ante accesos concurrentes?
- le uso concurrente
- ción se transfiere entre cliente y servidor?
- cceso
- guarda en cache y cómo se gestiona?
- cache

Localización de ficheros

- Resolución obtiene nombre interno (UFID)
- Uso de UFID con dir. máquina donde fichero está localizado
  - No proporciona, en principio, independencia de la posición
    - Nombre de fichero cambia cuando éste migra
- Uso de UFID que no contenga información de máquina:
  - Por ejemplo (AFS):
    id. único de volumen + id. inodo + nº versión
  - Permite migración de volúmenes
  - Requiere esquema de localización: Volumen → Máquina
- Posibles esquemas de localización:
  - Tablas que mantengan la información de ubicación (AFS)
  - Uso de *broadcast* para localizar nodo
- Uso de "cache de localizaciones" en clientes

Semánticas de uso concurrente

- Sesión: serie de accesos que realiza cliente entre *open* y *close*
- La semántica especifica el efecto de varios procesos accediendo de forma simultánea al mismo fichero
- Semántica UNIX
  - Una lectura *ve* los efectos de todas las escrituras previas
  - El efecto de dos escrituras sucesivas es el de la última
- Semántica de sesión (AFS)
  - Cambios a fichero abierto, visibles sólo en nodo que lo modificó
  - Una vez cerrado, cambios visibles sólo en sesiones posteriores
  - Múltiples imágenes simultáneas del fichero
  - Dos sesiones sobre mismo fichero que terminan concurrentemente:
    - La última deja el resultado final
  - No adecuada para procesos con acceso concurrente a un fichero

odelo de acceso

escarga
s completas del fichero
almacena en memoria o discos locales
utiliza semántica de sesión
as transferencias
n con mucha latencia

icio remoto
proporcionar todas las operaciones sobre el fichero
oques
/servidor

Fernando Pérez Costoya

Modelo carga/descarga

- Correspondencia petic. de aplicación y mens. de protocolo:
  - *open* → mensaje de descarga (*download*)
    - se realiza traducción y servidor envía fichero completo
    - cliente almacena fichero en cache local
  - *read/write/lseek* → no implica mensajes de protocolo
    - lecturas y escrituras sobre copia local
  - *close* → mensaje de carga (*upload*)
    - si se ha modificado, se envía fichero completo al servidor

Sistemas Distribuidos 18 Fernando Pérez Costoya

le serv. remoto: con estado

ia petic. de aplicación y mens. de protocolo:
aje de apertura
aducción
bilita zona de memoria para info. de la sesión
specífico para esa sesión
k → mensaje de protocolo correspondiente
luye id. sesión
tura y escritura incluye tamaño pero no posición
mación sobre la sesión almacenada en servidor (posición)
aje de cierre
ra zona de memoria de la sesión
o con estado (frente a sin estado):
pequeños, posibilidad de realizar políticas
n servidor (p.ej. lectura anticipada), procesamiento
posiblemente un poco más eficiente

Fernando Pérez Costoya

Modelo de serv. remoto: sin estado

- Correspondencia petic. de aplicación y mens. de protocolo:
  - *open* → mensaje de apertura
    - se realiza traducción
    - servidor no habilita zona de memoria para info. de sesión (cliente sí)
    - retorna id. interno del fichero UFID
  - *read/write* → mensaje de protocolo correspondiente
    - mensaje autocontenido
    - mensaje incluye UFID, tamaño y posición
  - *lseek* → no implica mensaje de protocolo
  - *close* → no implica mensaje de protocolo
    - cliente libera zona de memoria de la sesión
- Ventajas servicio sin estado (frente a con estado):
  - Tolerancia a fallos ante rearranque del servidor, posiblemente menos mensajes, no hay gastos de recursos en servidor por cada cliente (*escalabilidad*)

Sistemas Distribuidos 20 Fernando Pérez Costoya

stión de cache

ache permite mejorar el rendimiento

iples niveles de un SD:

servidores
s accesos a disco
clientes
áfico por la red
arga en los servidores
rse en discos locales (no permite nodos sin disco)
acidad pero más lento
il, facilita la recuperación
oria principal
apacidad pero más rápido
volátil

Fernando Pérez Costoya

Uso de cache en clientes

- Empleo de cache de datos en clientes
  - Mejora rendimiento y capacidad de crecimiento
  - Introduce problemas de coherencia
- Otros tipos de cache
  - Cache de nombres
  - Cache de metadatos del sistema de ficheros
- Políticas de gestión de cache de datos:
  - Política de actualización
  - Política de coherencia
- Enfoque alternativo: Cache colaborativa (xFS)
  - Si bloque solicitado está en la cache de otro cliente, se copia de ésta

Sistemas Distribuidos 22 — Fernando Pérez Costoya

ca de actualización

iata (*write through*)
d
son más lentas
ntación en la información transferida por la red
a (*delayed write*)
rápidas
áfico en la red
den borrarse antes de ser enviados al servidor
d
r los datos?
iódico
errar (*Write-on-close*)

Fernando Pérez Costoya

Coherencia de cache

- El uso de cache en clientes produce problema de coherencia
  - ¿es coherente una copia en cache con el dato en el servidor?
- Estrategia de validación iniciada por el cliente
  - cliente contacta con servidor para determinar validez
    - en cada acceso, al abrir el fichero o periódicamente
- Estrategia de validación iniciada por el servidor
  - servidor avisa a cliente (*callback*) al detectar que su copia es inválida
    - generalmente se usa *write-invalidate* (no *write-update*)
  - servidor almacena por cada cliente info. sobre qué ficheros guarda
    - implica un servicio con estado

Sistemas Distribuidos 24 — Fernando Pérez Costoya

ache: semántica de sesión

ada por el cliente (usada en AFS versión 1):
contacta con servidor enviando nº de versión (o fecha
del fichero almacenado en cache local (si lo hay)
rueba si corresponde con versión actual:
trario, se envia la nueva copia

ada por el servidor (usada en AFS versión 2):
n cache local, en apertura no se contacta con servidor
cena información de qué clientes tienen copia local
vuelca nueva versión del fichero al servidor:
vía invalidaciones a clientes con copia
le mensajes entre cliente y servidor
miento y *escalabilidad*
gestión de *callbacks*
fácilmente en modelo cliente-servidor clásico

Fernando Pérez Costoya

C. de cache: semántica UNIX

- Validación iniciada por el cliente
  - Inaplicable
    - Hay que contactar con servidor en cada acceso para validar info.
- Validación iniciada por el servidor. 2 ejemplos de protocolos:
  - *Prot1*: control en la apertura con desactivación de cache
    - Basado en Sprite: SOD desarrollado en Berkeley en los 80
  - *Prot2*: uso de *tokens*
    - Basado en DFS, sistema de ficheros distribuido de DCE (*Open Group*)

Sistemas Distribuidos
26

Fernando Pérez Costoya

he: semántica UNIX. *Prot1*

info. de qué clientes tienen abierto un fichero
urrente conflictivo (1escritor + otro(s) cliente(s))
y se usa acceso remoto en nodos implicados
contacta con servidor especificando:
so + nº de versión de copia en cache (si la hay)
icto de acceso:
el cliente en cache es más antigua, se indica que la invalide
roduce un conflicto de acceso:
a los clientes con el fichero abierto una orden de
y desactivación de la cache para ese fichero
n escritor se le pide un volcado previo
cliente que invalide y desactive la cache para ese fichero
e encuentra que ya hay conflicto
cliente que invalide y desactive la cache para ese fichero

Fernando Pérez Costoya

C. de cache: semántica UNIX. *Prot2*

- Para realizar operación se requiere *token* correspondiente
  - *Token* (de lectura o escritura) asociado a un rango de bytes
- Si cliente solicita operación y no está presente *token* requerido en su nodo, se solicita al servidor de ficheros
- Para una zona de un fichero, el servidor puede generar múltiples *tokens* de lectura pero sólo uno de escritura
- Si existen múltiples *tokens* de lectura y llega solicitud de escritura, servidor reclama los *tokens*
  - Cliente devuelve *token* e invalida bloques de cache afectados
  - Cuando todos devueltos, servidor manda *token* de escritura
- Si hay un *token* de escritura y llega solicitud de lectura o escritura, servidor reclama el *token*:
  - Cliente vuelca e invalida bloques de cache afectados

Sistemas Distribuidos
28

Fernando Pérez Costoya

n estado basado en *leases*

X requiere servicio con estado
servicio con estado pero con buena recuperación?
ón con plazo de expiración
varse
os servicios además de la coherencia
token tiene un plazo de expiración
o cliente considera que *token* ya no es válido
er a solicitarlo
servidor con estado pero fácil recuperación:
rranca servidor no entrega *tokens* hasta que haya pasado
piración con lo que todos los *tokens* están caducados

Fernando Pérez Costoya

Gestión de cerrojos

- SFD ofrecen cerrojos de lectura/escritura
  - múltiples lectores y un solo escritor
- Peticiones *lock*/*unlock* generan mensajes correspondientes
  - *lock*: si factible retorna OK; sino no responde
  - *unlock*: envía a OK a cliente(s) en espera
- Requiere un servicio con estado:
  - servidor almacena qué cliente(s) tienen un cerrojo de un fichero y cuáles están en espera
- Problema: cliente con cerrojo puede caerse
  - Solución habitual: uso de *leases*
  - Cliente con cerrojo debe renovarlo periódicamente

Sistemas Distribuidos 30 — Fernando Pérez Costoya

File System (NFS) de Sun

e un protocolo para acceso a ficheros remotos
do para entornos heterogéneos
-1813 (descrita en esta presentación)
-3010 (última versión; cambios en arquitectura)
gracias al uso de RPC/XDR de ONC
da en "RPC segura"
áquina monta directorio remoto en SF local
nbres es diferente en cada máquina
migración ni replicación
protocolos: montaje y acceso a ficheros

Fernando Pérez Costoya

Protocolo de montaje

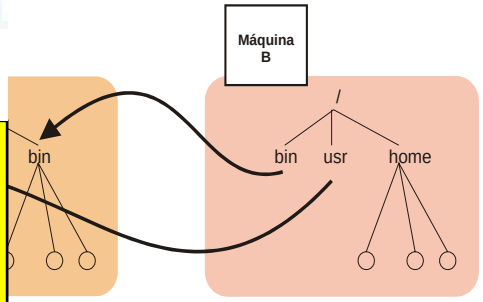
- Establece una conexión lógica entre el servidor y el cliente
- Cada máquina incluye una "lista de exportación"
  - qué "árboles" exporta y quién puede montarlos
- Petición de montaje incluye máquina y directorio remotos
  - Se convierte en RPC al servidor de montaje remoto
  - Si permiso en lista, devuelve un identificador "opaco" (*handle*)
    - Cliente no conoce su estructura interna
- La operación de montaje sólo afecta al cliente no al servidor
  - se permiten montajes NFS anidados, pero no "transitivos"
- Aspectos proporcionados por algunas implementaciones:
  - montajes *hard* o *soft*: en montaje, si servidor no responde...
    - espera ilimitada (*hard*) o plazo máximo de espera (*soft*)
  - automontaje: no solicita montaje hasta acceso a ficheros

Sistemas Distribuidos 32 — Fernando Pérez Costoya

lo de montado en NFS

orta /usr y /bin

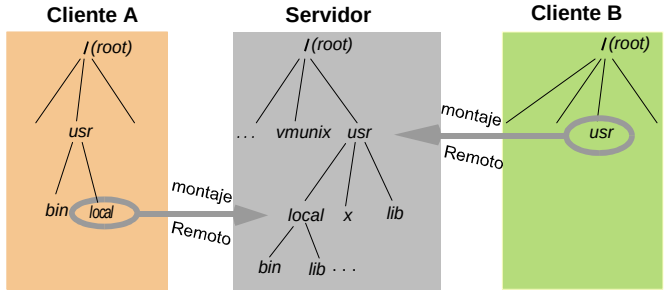
A:/usr /usr



Fernando Pérez Costoya

Ejemplo de montado en NFS

- Imagen diferente del sistema de ficheros



Fernando Pérez Costoya

Protocolo NFS

ara realizar operaciones sobre ficheros remotos
un fichero en un directorio (LOOKUP)
radas de directorio
de enlaces y directorios
tributos de un fichero
tura. En vers. 2 escritura síncrona en disco de servidor
ermite asíncrona (COMMIT fuerza escritura en disco)
sin estado (no en versión 4)
utocontenidas
zado por LOOKUP (no hay CLOSE)
ativa componente a componente
andle de directorio, fichero) → *handle* de fichero
ofrece mecanismos de control de concurrencia
endiente de NFS: *Network Lock Manager*

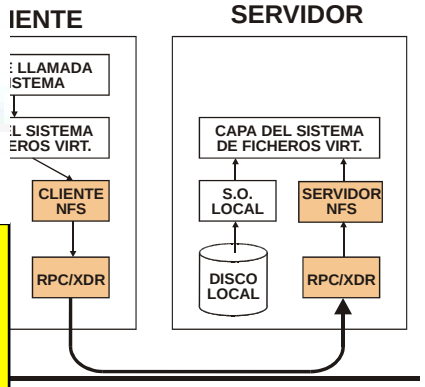
Fernando Pérez Costoya

Implementación Sun/NFS

- Arquitectura basada en sistema de ficheros virtual (VFS)
- *Vnodo* apunta a un nodo local o a uno remoto (*Rnodo*)
- Cada *Rnode* contiene *handle* del fichero remoto
- Contenido del *handle* depende de sistema remoto
- En sistemas UNIX se usa un *handle* con tres campos:
  - id. del sistema de ficheros
  - número de inodo
  - número del versión del inodo (se incrementa en cada reutilización)
- En montaje se obtiene *handle* de la raíz del subárbol montado
- Posteriores operaciones *lookup* obtienen sucesivos *handles*

Fernando Pérez Costoya

tectura de Sun/NFS



Fernando Pérez Costoya

Acceso a ficheros en Sun/NFS

- Las transferencias se realizan en bloques de 8 KB
- Los bloques se almacenan en la cache de los clientes
- Los clientes realizan lecturas adelantadas de un bloque
- Las escrituras se realizan localmente.
- Los bloques se envían al servidor cuando se completan o se cierra el fichero
- Cache del servidor:
  - escritura síncrona o asíncrona según lo indicado por el cliente
- 3 tipos de cache en el cliente:
  - cache de nombres para acelerar las traducciones
  - cache de atributos de ficheros y directorios (fechas, dueño, ...)
  - cache de bloques de ficheros y directorios

Sistemas Distribuidos 38 — Fernando Pérez Costoya

cia de cache en Sun/NFS

guna semántica

da por el cliente:

n sobre un fichero devuelve sus atributos

indican que el fichero se ha modificado

los datos del fichero en cache de bloques

ache de bloques y atributos tienen un tiempo de vida

en es ese periodo se descartan

:

para ficheros

ctorios

Fernando Pérez Costoya

Novedades de la versión 4 de NFS

- Servicio con estado basado en *leases* (hay OPEN y CLOSE)
- Diseñado para ser usado en Internet
- Integra protocolo de montaje y de cerrojos
  - Un solo puerto fijo (2049): facilita atravesar cortafuegos
- Empaquetamiento de operaciones:
  - Una llamada RPC (COMPOUND) con múltiples operaciones
- Operación LOOKUP puede resolver el camino completo
  - Montajes en el servidor visibles por el cliente
- Uso de listas de control de acceso (ACL)
- Atributos del fichero incluye un mecanismo de extensibilidad

Sistemas Distribuidos 40 — Fernando Pérez Costoya

ew File System (AFS)

do en Carnegie- Mellon (desde 1983)
versión AFS-2
oducto de Transarc (incluida en IBM)
sión de libre distribución para UNIX y Windows
uidos a gran escala (5000- 10000 nodos)
nodos cliente y servidores dedicados
nte tienen que tener disco
s dos espacios de nombres:
tido (directorio `/afs`)
ólo para ficheros temporales o de arranque
ionan el espacio compartido
todos los clientes del espacio compartido

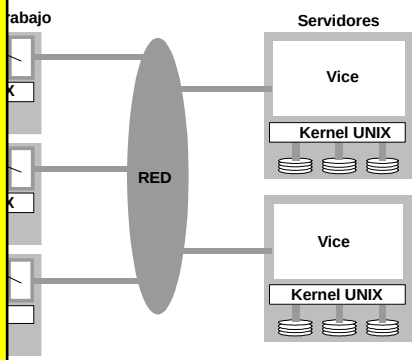
Fernando Pérez Costoya

Estructura de AFS

- Dos componentes que ejecutan como procesos de usuario
- *Venus*:
  - ejecuta en los clientes
  - SO le redirecciona peticiones sobre ficheros compartidos
  - realiza las traducciones de nombres de fichero
    - resolución dirigida por el cliente
    - cliente lee directorios: requiere formato homogéneo en el sistema
- *Vice*:
  - ejecuta en los servidores
  - procesa solicitudes remotas de clientes
- Usan sistema de ficheros UNIX como almacén de bajo nivel

Sistemas Distribuidos 42 Fernando Pérez Costoya

tructura de AFS



Fernando Pérez Costoya

Espacio de nombres compartido

- Los ficheros se agrupan en *volúmenes*
  - Unidad más pequeña que un sistema de ficheros UNIX
- Cada fichero tiene identificador único (UFID: 96 bits)
  - Número de volumen
  - Número de *vnodo* (dentro del volumen)
  - Número único: permite reutilizar números de vnodo
- Los UFID son transparentes de la posición
  - un volumen pueden cambiar de un servidor a otro.
- Soporte a la migración de volúmenes
- Estrategia de localización
  - número de volumen → servidor que lo gestiona
  - tabla replicada en cada servidor
  - cliente mantiene una cache de localización
    - si falla repite proceso de localización

Sistemas Distribuidos 44 Fernando Pérez Costoya

cceso a ficheros

a/descarga
idor transfiere fichero completo al cliente
: fragmentos de 64Kbytes
a el fichero en la cache local de los clientes
sco local (la cache es no volátil)
ras localmente sin intervenir *Venus*
X opera aunque de manera transparente a AFS
ceso cierra un fichero (*close*)
cado se envía al servidor (*write-on-close*)
n cache local para futuras sesiones
de directorios y atributos directamente al

Fernando Pérez Costoya

Coherencia de cache (1/2)

- Semántica de sesión
- Validación iniciada por servidor basada en *callbacks*
- Cuando cliente abre fichero del que no tiene copia local (o no es válida), contacta con el servidor
  - el servidor "anota" que el fichero tiene un *callback* para ese cliente
- Siguientes aperturas del fichero no contactan con servidor
- Cuando cliente cierra un fichero que ha modificado:
  - Lo notifica y lo vuelca al servidor
  - Servidor avisa a los nodos con copia local para que la invaliden:
    - Eevoca el *callback*
  - Solicitud en paralelo usando una multiRPC

Sistemas Distribuidos
46
Fernando Pérez Costoya

encia de cache (2/2)

na revocación a un nodo:
fichero abierto continúan accediendo a copia anterior
a cargará el nuevo contenido desde el servidor
AFS asumen que los datos en su cache son
s no se notifique lo contrario
acena por cada fichero una lista de clientes
ia del fichero en su cache:
a todos los clientes que tienen copia y no sólo a los
erto el fichero

Fernando Pérez Costoya

Coda

- Descendiente de AFS orientado a proporcionar alta disponibilidad mediante replicación de volúmenes
- Lectura de cualquier copia – Escritura en todas
- Si red "partida": cliente sólo actualiza copias accesibles
  - Se mantienen contadores de versión en cada copia de fichero
- En reconexión: se comparan contadores de las copias
  - Si no conflicto → se propaga a todas las copias la última versión
  - Si conflicto → reconciliación automática o manual
    - Ejemplo de automática: reconciliación de directorio
- Permite operación desconectada del cliente
  - Usuario puede sugerir qué ficheros deberían estar en cache
  - En reconexión: conciliación entre cache del cliente y servidores
  - Aunque no concebido para ello, es aplicable a computación móvil

Sistemas Distribuidos
48
Fernando Pérez Costoya