Ejemplos de sistemas de ficheros distribuidos

Joaquín Seoane Pascual
joaquin@dit.upm.es

Departamento de Ingeniería de Sistemas Telemáticos
Universidad Politécnica de Madrid

8 de noviembre de 2001

Índice General

Ejemplos

NFS SUN (1985)

- Amplia difusión (dominio público), incluso MS/DOS, Mac.

SMB IBM/Microsoft

- Inevitable.

AFS CMU – Transarc – IBM (1986)

- Escalabilidad.
- Seguridad.
- Patrones de uso.

CODA CMU (1990)

- Sucesor de AFS.
- Replicación.
- Operación desconectada.

NFS

- Extensión inexacta de la semántica unix:
  - Mismas operaciones.
  - Montaje remoto.
- Directorio integrado.
- Protocolo con operaciones idempotentes.
- Caches en memoria de cliente y servidor.
- Replicación (sólo lectura) con *automounter*.
- Escalabilidad limitada.
- Difícil de mantener.
- Gestor de concurrencia independiente.

El montaje en Unix

- Diseñado para integrar volúmenes en jerarquía de ficheros.

```
mount /dev/fd0 /mnt
ls /mnt
```

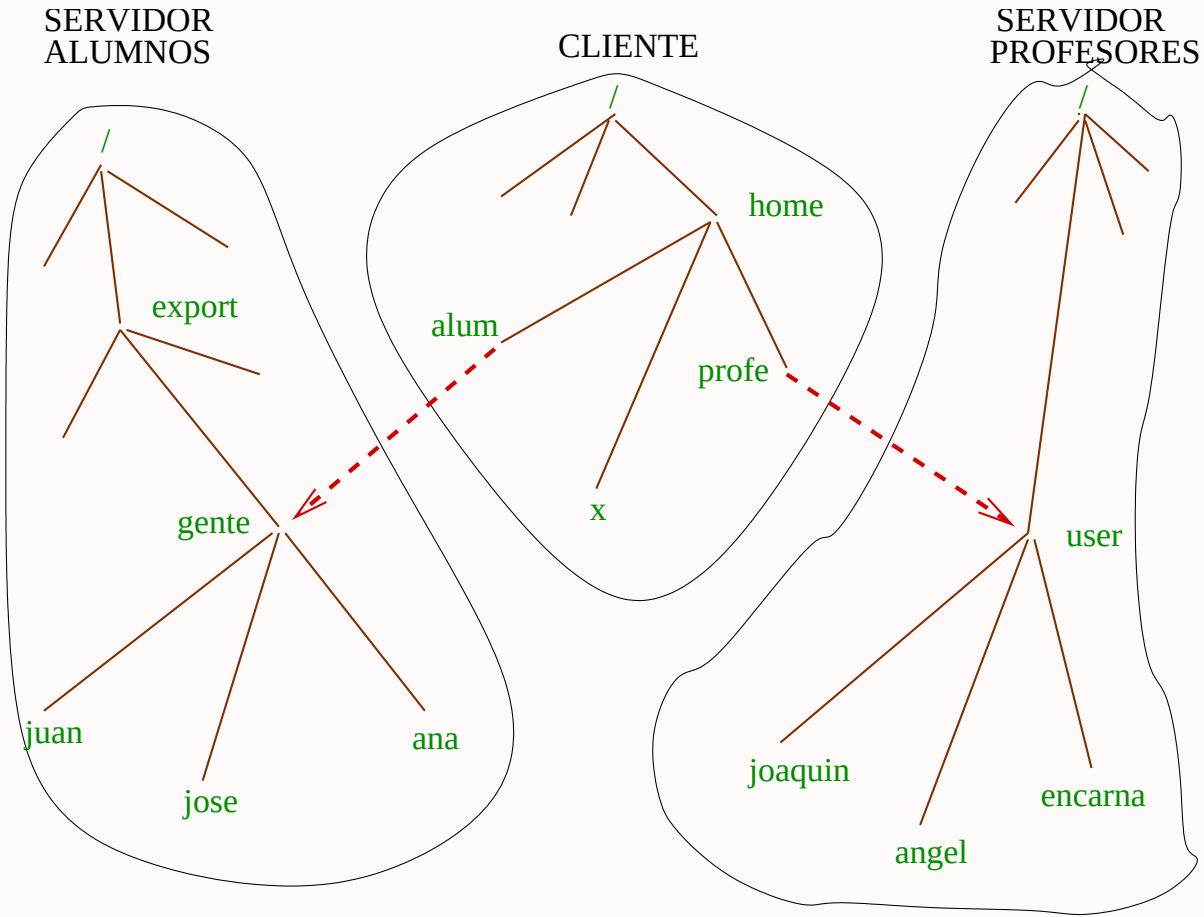
- Es una operación privilegiada y global.
- Suele haber opciones:
  - Sólo lectura.
  - No ejecución (o no ejecución *setuid*).
  - Escrituras síncronas.

```
mount -oro,nosuid /dev/fd0 /mnt
```

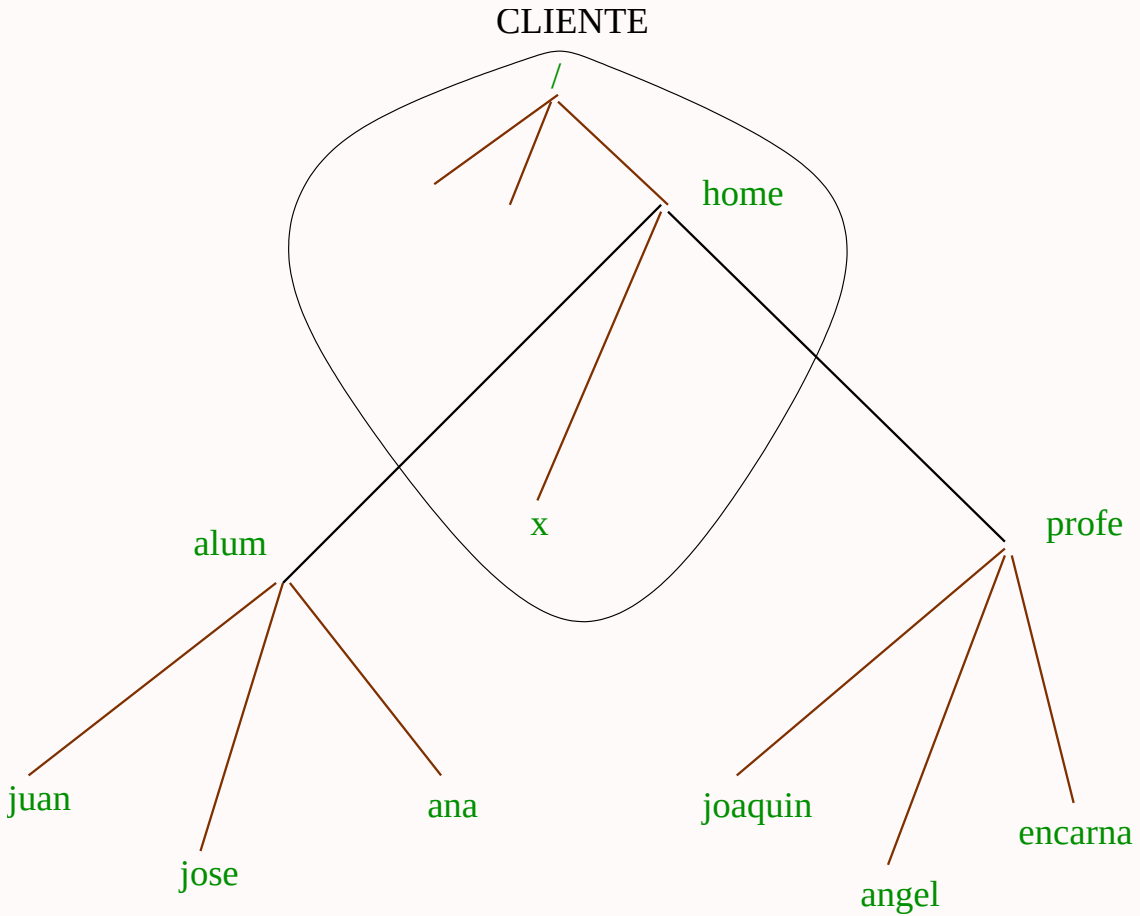
- A veces soportan heterogeneidad:

```
mount -t msdos /dev/fd0 /mnt
```

Montaje remoto



Montajes hechos



Operaciones de montaje remoto

```
mount serv1:/export/gente /home/alum
mount serv2:/user /home/profe
```

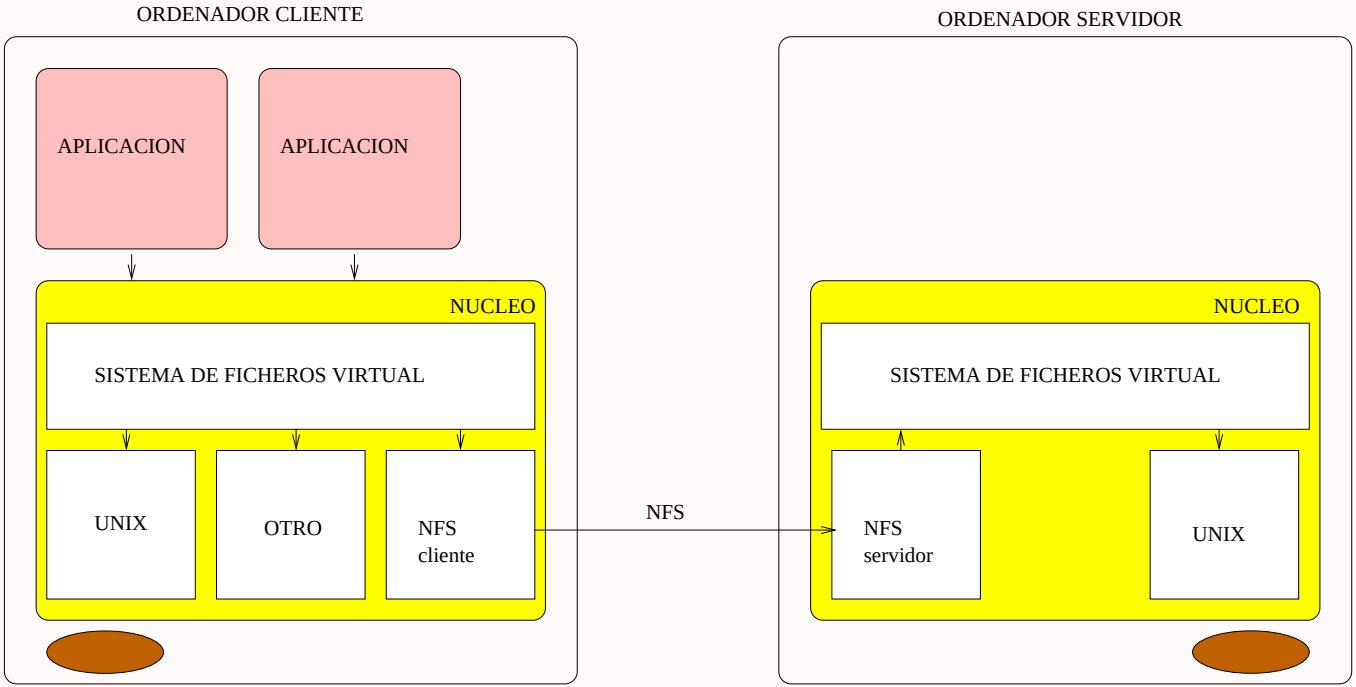
- Todos pueden ser clientes y servidores.
- Puede haber varias vistas del espacio de nombres.
- Se pueden reubicar directorios, pero es muy complejo:
  → Cambiar tablas de montaje en todos los clientes.

Montaje inicial

Fichero `/etc/fstab`:

```
/dev/hda1      /     ext2 defaults,errors=remount-ro 1 1
/dev/hda2      /usr  ext2 defaults                   1 2
cuentas:/home /home nfs  rsize=8192,wsize=8192       0 0
info:/pub/doc /doc  nfs  ro,rsize=8192               0 0
```

Realización: arquitectura NFS



Realización: el sistema de ficheros virtual

- Intercepción de todas las llamadas de ficheros (módulo *VFS*).
- El apertura, las tablas de montaje determinan si local o remoto.
  - Si local o remoto (y dónde).
  - Tipo/protocolo y parámetros.
  - Se anota todo en un *v-node*.
- Resto de operaciones dirigidas por *v-node*.
- Si remoto, se traduce a llamadas remotas sin estado al servidor.
- En red se usa un *file handle* (UFID):

sistema de ficheros	número de i-nodo	número de generación

- El servidor debería ser un conjunto de hebras en el núcleo (a veces en espacio de usuario).

Los procedimientos remotos de NFS

- `rpc` + `xdr` de Sun.
- Usados en otros servicios.
- Programas numerados asociados a puertos UDP.
- La asociación la hace el programa `portmapper`, que siempre escucha en el puerto 111.

```
rpcinfo -p servidor      # consulta registros
rpcinfo -u servidor nfs  # llama al procedimiento 0
```

El protocolo de procedimientos remotos

$read(fh, pos, cantidad) \rightarrow atrib, datos$	lectura sin estado
$write(fh, pos, cantidad, datos) \rightarrow atrib$	escritura sin estado
$getattr(fh) \rightarrow atrib$	lectura de atributos
$setattr(fh, atrib) \rightarrow atrib$	cambio de atributos
$lookup(dirfh, nombre) \rightarrow fh, attr$	búsqueda
$create(dirfh, nombre, atrib) \rightarrow fh, atrib$	creación
$remove(dirfh, nombre)$	borrado
$rename(dirfh, nombre, nuevodirfh, nuevonombre)$	renombrar
$link(nuevodirfh, nuevonombre, dirfh, nombre)$	enlace
$link(nuevodirfh, nuevonombre, cadena)$	enlace simbólico
$readlink(fh) \rightarrow cadena$	leer enlace
$readdir(dirfh, galleta, num) \rightarrow entradas, galleta$	leer directorio
$statfs(fh) \rightarrow estadosf$	sistema de ficheros

Control de acceso y autenticación

- El servidor exporta selectivamente (`/etc/exports`).

```
/usr *.dit.upm.es(ro)
/home *.dit.upm.es(rw)
/home/joaquin viracocha.gbt.tfo.upm.es(rw)
/home/ftp/pub (ro)
```

- Los clientes envían credencial y autenticador en cada operación remota.
  - Ordinaria (*uid*, *gid*, *puerto reservado*).
  - Claves públicas (Diffie-Hellmann) + DES.
  - Kerberos.

El servicio de montaje

- Independiente del protocolo de acceso.
- Lee `/etc/exports`.
- Recibe un *nombre de directorio* y devuelve un *file handle*.
- Conserva cierto estado (operaciones de *desmontar*, *listar montados*).
- Puede colaborar en la autenticación.

Opciones de montaje remoto

- El cliente puede especificar:
  - Montaje *duro* o *blando* (se puede rendir).
  - Protocolo (*tcp*, *udp*), *puerto*, ...
  - Parámetros: tamaño de mensaje, plazos, números de retransmisiones, ...

```
mount -orsize=8192,wsize=8192,timeo=10,retrans=5 \
      jungla.dit.upm.es:/home /home
```

Traducción de rutas

- Se pueden hacer montajes en directorios remotos.
- El servidor no conoce esos montajes.
- Cada tramo de una ruta puede estar en un servidor (o local).
- La apertura se hace por pasos en el cliente:
  → operaciones `lookup(dirfh, nombre)`.
- Estas operaciones son muy abundantes (conviene *cachear*).

Automontador

- Evitar tablas de montaje grandes.
- Evitar depender de muchas maquinas al arrancar.
- Tener la posibilidad de usar servidores alternativos (sólo lectura):
  - Tolerancia a fallos.
  - Balance de cargas.
- Servidor local:
  - Atiende determinados directorios.
  - Sólo implementa `lookup`, `readdir`.
  - Monta bajo demanda (en otro directorio).
  - Desmonta si no hay actividad.
  - Muestra un enlace simbólico al sitio montado.

Caches de servidor

- Las mismas que para el sistema de ficheros local.
  - Carga anticipada.
  - Escritura retrasada.
- Las escrituras de NFS no deben retrasarse:
  - Prever la caida del servidor inadvertida por el cliente.
  - Escribir y luego responder.
  - Evitable con baterías en la memoria.
  - Operación `commit` (NFS-3).

Caches de clientes

- Producen incoherencia si hay varios clientes.
- Mejoran el rendimiento.
- Compromiso:
  - Todo bloque leído se *guarda* con atributos:
    - última validación.
    - última modificación en el servidor.
  - Todo bloque *fresco* se usa sin contacto con el servidor.
  - Todo acceso a bloque *viejo* solicita atributos y datos (si se ha modificado). Puede invalidar otros bloques.
  - La *frescura* es parámetro de montaje y adaptativo (eg: 3 segundos para ficheros, 30 segundos para directorios).
  - Las operaciones siempre se traen los atributos.

Escritura y caches de clientes

- Las escrituras marcan páginas como sucias.
- Las escrituras sólo se hacen cuando es imprescindible (cierre, `sync`, falta de sitio, ...).
- Mejoras con *bio-daemons* (vaciados y carga anticipada).

Resumen de NFS

Transparencia de acceso	Misma API.
Transparencia de ubicación	Nombre independiente de ubicación. Conviene independencia de cliente.
Transparencia de migración	Rehacer tablas de montaje.
Escalabilidad	Caches. Limitada por sondeo de atributos.
Replicación	Sólo lectura con automontador.
Heterogeneidad	Protocolo RPC/XDR.
Tolerancia a fallos	Idempotencia, escrituras comprometidas.
Consistencia	Límites temporales.
Seguridad	Credenciales autenticables (Kerberos, RFC 2203).

AFS

- Diseñado para 10.000 estaciones en entorno inseguro.
- Facilidad de administración.
- Patrones de uso (académico, desarrollo de programas).

La mayoría de los ficheros < 10 k.
Seis veces más lecturas que escrituras.
Acceso secuencial.
Lectura completa.
Gran mutabilidad.
Compartición muy rara (en escritura más).
No personalizar las estaciones.

Principios de diseño.

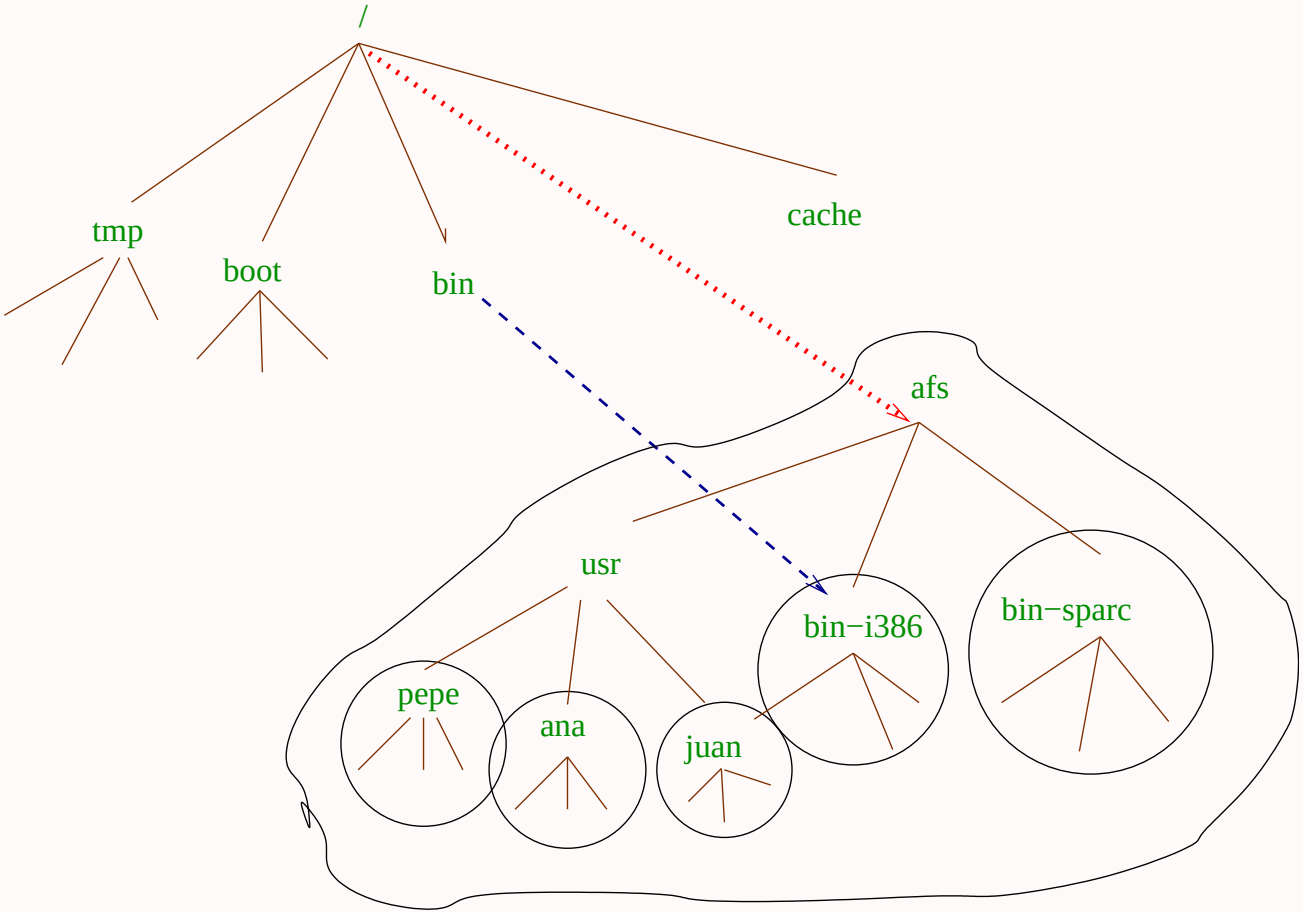
- Caches en disco:
  - De ficheros completos (AFS-1 y AFS-2).
  - De porciones grandes (64k en AFS-3).
- Protocolo de transferencia masiva.
- Se trae al abrir (si no está o la copia no es válida).
- Se copia al servidor al cerrar, **sólo si se ha modificado**.
- El servidor avisa de las modificaciones.
- Al arrancar el cliente, debe validar todos los ficheros de la cache.
- El sistema sólo intercepta `open` y `close`.
- Servidores dedicados.
- Ficheros agrupados en volúmenes.
- Administración por células (AFS-3).

Espacio de nombres

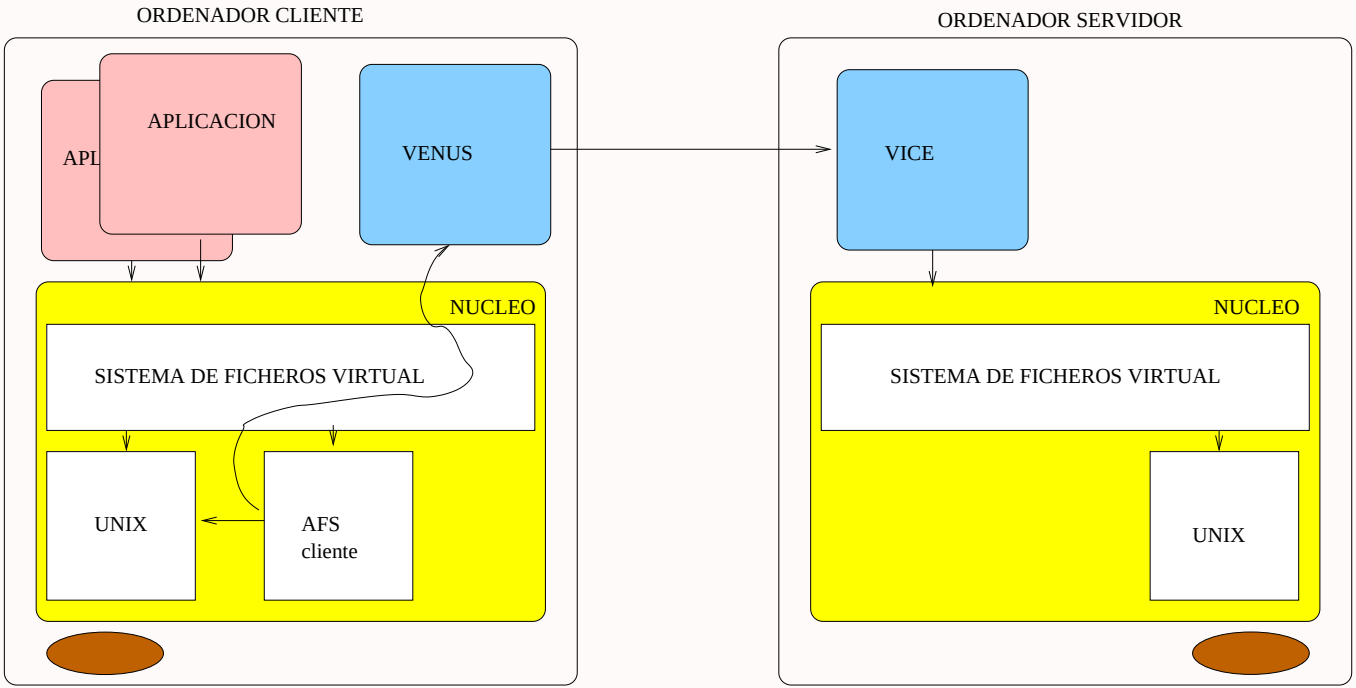
- Los volúmenes se montan en una operación **global**.
- Información de ubicación y montaje replicada.
- El identificador de fichero contiene el volumen:

número de volumen	número de fichero	uniquificador

Espacio de nombres



Arquitectura de AFS



Realización de primitivas en AFS

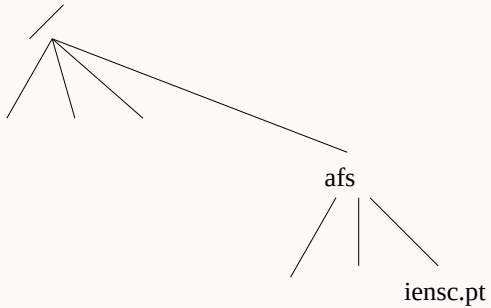
Proceso de usuario	*VFS*	*Venus*	*Red*	*Vice*
`open(f, modo)`	Si `f` remoto, pasa la petición a *Venus*. Abre el fichero local y devuelve descriptor `fd`.	Si no hay copia local válida, pasa la petición a *Vice*. Pone la copia en disco local.	⇒ ⇐	Transfiere copia y promete aviso de invalidación.
`read(fd, b, l)`	Lectura normal.			
`write(fd, b, l)`	Escritura normal.			
`close(fd)`	Cierre normal de copia local y, si modificada, notifica a *Venus*.	Manda copia a *Vice*.	⇒	Reemplaza el fichero y avisa a los clientes a los que haya prometido avisar.

El protocolo de procedimientos remotos

$Fetch(fh) \rightarrow atrib, datos$	lectura con promesa
$Store(fh, atrib, datos)$	escritura con promesa y aviso
$Create() \rightarrow fh$	creación con promesa
$Remove(fh)$	borrado
$SetLock(fh, modo)$	cierre compartido o exclusivo (expira 30 s)
$ReleaseLock(fh)$	apertura de cerrojo
$RemoveCallback(fh)$	vaciado de cache
$BreakCallback(fh)$	retrollamada de invalidación
.....	Operaciones de directorio, administrativas, etc.

Otras características

- Trasmisión cifrada.
- Listas de acceso.
- Movilidad dinámica de volúmenes.
- Respaldos dinámicos:
  - Copia del volumen congelada (*copy on write*).
  - Paso a cinta de la copia congelada.
- Integración de volúmenes de otras células.

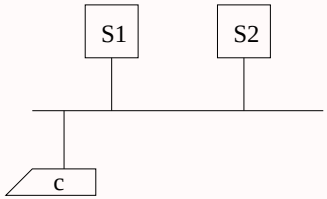


- Replicación de lectura.

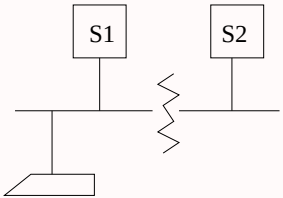
CODA

- Conservar ventajas de AFS.
- Tolerancia a fallos:
  - De servidor.
  - De red.
  - Compromiso entre autonomía e interdependencia.
- Soporte de portátiles:
  - Aislamiento = fallo deliberado.
  - Asitencia manual a la cache.
  - Soporte de conexiones de baja calidad.
    - Teléfono, radio-paquete, infrarrojos...
- Servidores perezosos.

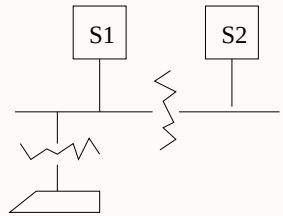
Transiciones



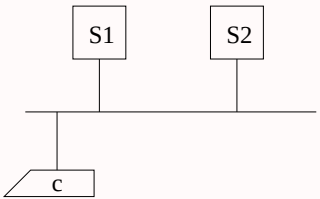
CONECTADO
TOTALMENTE



PARTICION



DESCONEXION



RECONEXION

Manejo de réplicas

- Réplicas de volumen → VSG.
- Réplicas accesibles → AVSG
  Si AVSG vacío → desconectado.

Replicación optimista

- Trabajar con la mejor copia accesible.
- Resolver inconsistencias en cuanto sea detectable:
  - Manual.
  - Automático.

Identificación de inconsistencias

Grabar, en una **transacción atómica**, la **historia de modificaciones**, junto al fichero.

Modificación $\longrightarrow$ identificador único	$m_1\ m_2\ m_3\ m_4\ m_5$	Dominante.
	$m_1\ m_2\ m_3$	Sumisa.
	$m_1\ m_2\ m_6$	Inconsistente.

- Actualizar copias sumisas.
- Resolver inconsistencias antes de actualizar.

Vectores de versiones

- Las historias crecen.
- Basta contar número de modificaciones vistas en cada réplica:

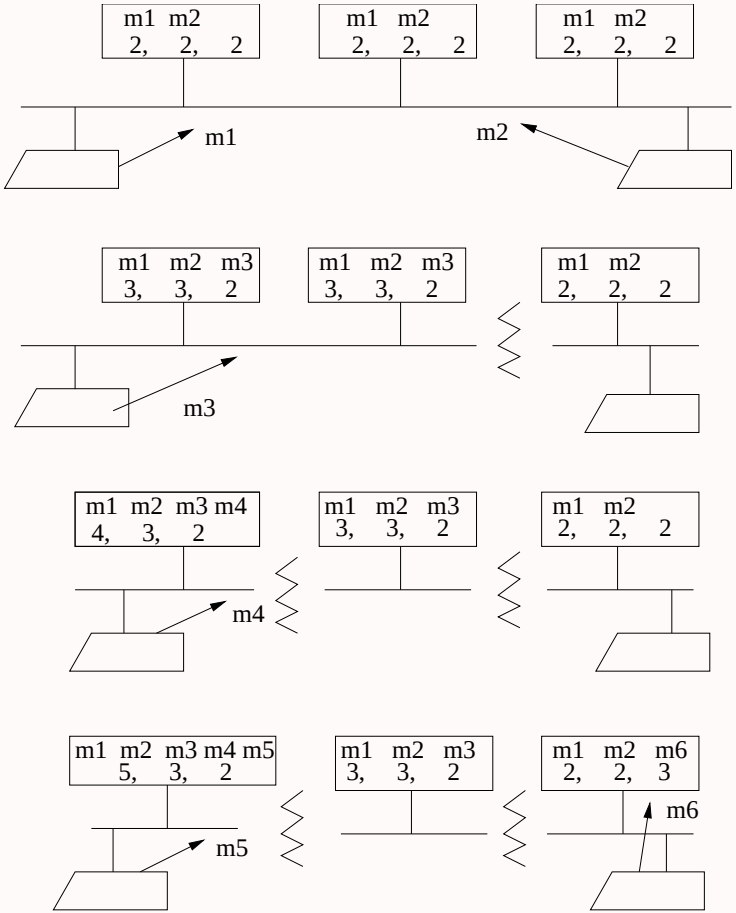
  Si se ha hecho una modificación $\rightarrow$ se hicieron las anteriores.

- Similar a los vectores de tiempo para caracterizar la causalidad.

  Consistente: Una copia tiene todos los contadores mayores o iguales que la otra.

  Inconsistente: Algún contador mayor y alguno menor.

Ejemplo: historias y vectores



Lectura de ficheros (en apertura)

- Si cache válida $\longrightarrow$ usarla.
- Si cache invalidada $\longrightarrow$
  - Lectura del preferido.
  - Obtención de los CVV.
  - Determinación de:
    - Si hay inconsistencias $\rightarrow$ resolución automática o manual.
    - Si hay sumisos $\rightarrow$ reemplazo.
    - Si preferido obsoleto $\longrightarrow$
      - cambio de preferido.
      - recarga.
      - invalidar volumen (posible pérdida de *callbacks*).

Escritura de ficheros (en cierre)

- Si no modificado $\longrightarrow$ nada.
- Si modificado $\longrightarrow$
  - Multicast a AVSG de fichero + CVV anterior.
  - Cliente actualiza CVV con los que responden.
  - Cliente informa de nuevo CVV (validando la escritura).
  - Actualización en *background* (segundo plano).
  - Puede haber conflicto.
  - *Callback* de servidores preferidos a sus clientes con copias.

Mantenimiento de AVSG

- Sondeo a VSG cada $\approx 10\,min$:
  - Ampliación $\longrightarrow$ invalidar.
  - Reducción $\longrightarrow$
    - No incluye preferido $\longrightarrow$ nada.
    - Incluye preferido $\longrightarrow$ invalidar.

Servidor preferido no está en AVSG de otro cliente

Pierdo los avisos de modificaciones $\rightarrow$ mantenimiento de CVV de volumen

- Examen en sondeos.
- Invalidar si hay posible incoherencia.

Funcionamiento desconectado

- Asistir a la cache, para que tenga lo que se va a necesitar o no lo elimine.
- Los objetos mutados no se borran de la cache.
- Se hace un registro de la actividad durante la desconexión, para reproducir al conectar.

Experiencia

- Rendimiento similar a AFS.
- Bastan $\approx 60\ MB$ de disco.
- $\approx 99'5\%$ de las modificaciones por el mismo usuario.
- Probabilidad de que los usuarios modifiquen el mismo objeto en una semana $\approx 0.4\%$.