CODA FileSystem

Sistema de ficheros distribuido

Daniel Rodríguez García
Alberto Peraleda de la Llave

ÍNDICE

1 OBJETIVOS Y DESCRIPCIÓN DEL SERVICIO

Coda es un sistema de ficheros distribuido, desarrollado en la Universidad Carnegie Mellon. Coda está basado en una arquitectura cliente/servidor, y ha sido diseñado para proporcionar funcionalidades no existentes en otros sistemas de ficheros distribuidos – por ejemplo NFS – como es el soporte para la movilidad de los clientes.

Las características principales de Coda (según describen los propios desarrolladores) son:

- Permite operar a los clientes *offline*, esto es, desconectados del sistema, de manera que pueden realizar cambios locales en los ficheros que se propagan a todo el sistema de ficheros una vez que el cliente en cuestión se reconecta al mismo.

- Facilita características para mejorar la resistencia a fallos en el sistema, mediante la posibilidad de incluir servidores con réplicas de los datos del sistema, y mecanismos para manejar conflictos entre servidores, para manejar fallos en la red, y para controlar la desconexión de los clientes y los posibles conflictos e inconsistencias que ello puede acarrear. También incorpora mecanismos de *backup* de los datos del sistema.

- Proporciona mejoras en el rendimiento, con respecto a otros sistemas de ficheros distribuidos, mediante la utilización de caches locales en los clientes y para la escritura de datos en los servidores.

- Incorpora características de seguridad basadas fundamentalmente en la autenticación de usuarios, mediante la utilización de listas de control de acceso (ACL's).

- El código está disponible de forma libre.

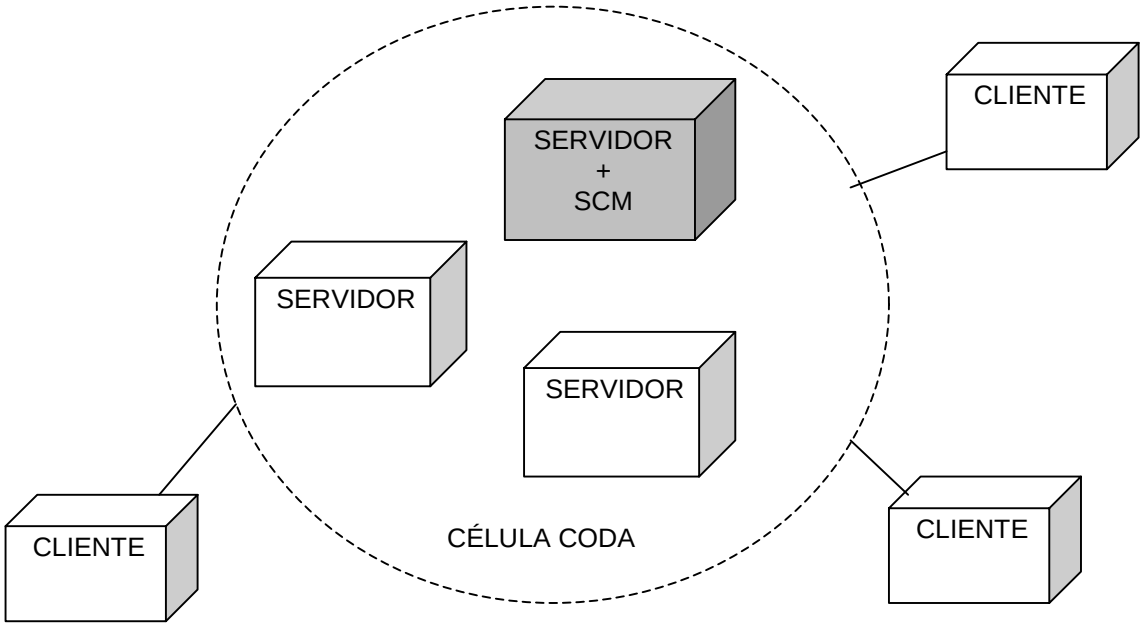
De esta forma, el objetivo principal perseguido con Coda es obtener un sistema de ficheros distribuido con un rendimiento mejorado, menos sensible a posibles indisponibilidades en los servidores o a sobrecargas de la red. Relacionada con esta meta está la provisión de cierto grado de movilidad a los clientes del sistema.

Actualmente existen versiones de Coda para diferentes variantes de Unix (incluido Linux Debian y RedHat) y Windows 95. La versión de Coda descrita en este documento es la 5.3.17-1.

2 ARQUITECTURA DEL SERVICIO

2.1 DESCRIPCIÓN GENERAL

Como ya se ha citado anteriormente en este documento, Coda está basado en una arquitectura cliente/servidor, como se muestra en la figura siguiente:



Los servidores de Coda se agrupan en conjuntos denominados células Coda. Un cliente sólo puede acceder a una célula Coda. Dentro de cada célula, siempre existe un servidor especial denominado SCM (*System Control Machine*), que es un servidor Coda normal, pero con la funcionalidad añadida de controlar todas las bases de datos de configuración de la célula. El SCM actúa como servidor de configuración para el resto de los servidores Coda de la célula (que actúan como clientes del SCM, desde el punto de vista de la configuración).

El sistema de ficheros Coda provee un mecanismo de identificación (y permisos) propio, independiente del proporcionado por el sistema operativo. Para operar con los permisos adecuados, los usuarios deben autenticarse en el sistema Coda.

2.2 REPRESENTACIÓN DE LA INFORMACIÓN

Los datos ofrecidos por los servidores Coda se agrupan en estructuras denominadas volúmenes. Un volumen tiene una estructura de directorios (como un sistema de ficheros de Unix), y sirve para representar un conjunto de los ficheros contenidos en un servidor Coda.

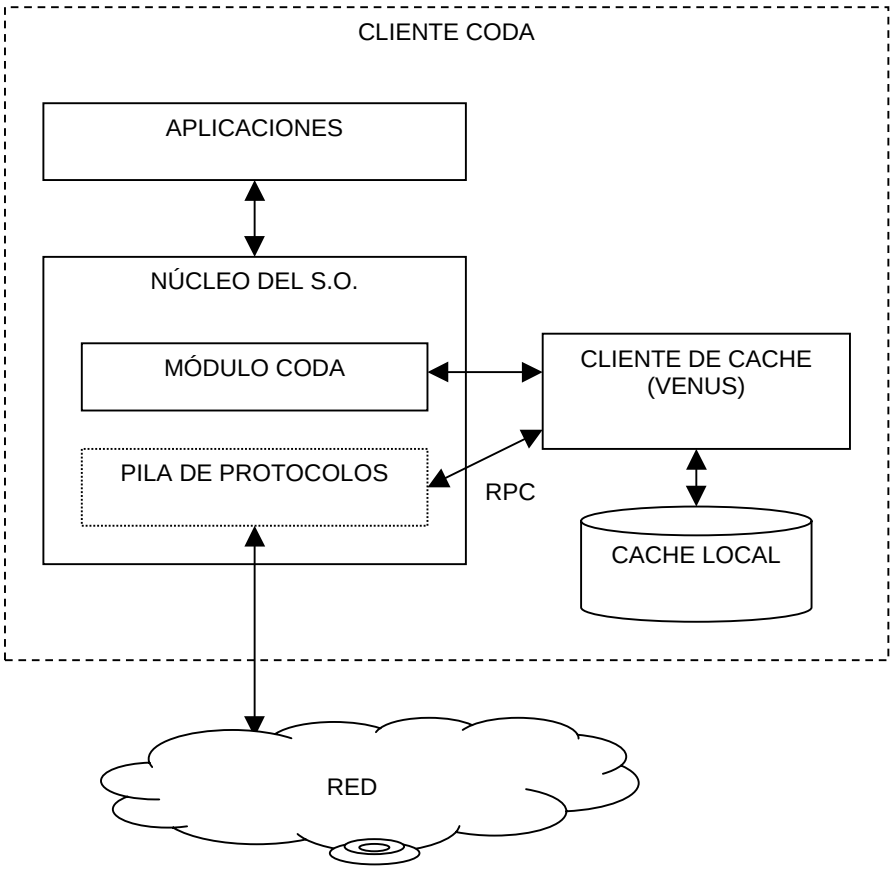
La información sobre los volúmenes se guarda dentro del servidor Coda como “metadatos” en un área de almacenamiento especial (una partición separada, o un fichero). Los ficheros asociados a un volumen se almacenan en el servidor de forma convencional (en su sistema de ficheros local).

Los volúmenes se pueden montar en el sistema de ficheros Coda, dentro de otros volúmenes ya montados. En una célula, existe siempre un volumen raíz que aparece montado por defecto en el directorio `/coda` de los clientes. Dentro de éste pueden montarse otros volúmenes, que aparecen como subdirectorios nuevos dentro del sistema de ficheros de Coda (en el nivel especificado para realizar el montaje: p.ej. en `/coda/varios/` se puede montar un nuevo volumen que aparecería como `/coda/varios/nombre_dir_nuevo_vol/`).

Una característica interesante de Coda es la "replicación" de volúmenes: un mismo volumen puede almacenarse en varios servidores; si al acceder un cliente al volumen un servidor falla, el sistema automáticamente recupera la información de otro de los servidores disponibles con ese volumen. El sistema propaga automáticamente los cambios realizados en el volumen hacia el servidor que ha fallado, cuando éste vuelve a estar en servicio.

2.3 ESTRUCTURA DE LOS CLIENTES CODA

La estructura interna de un cliente se muestra en la figura siguiente:

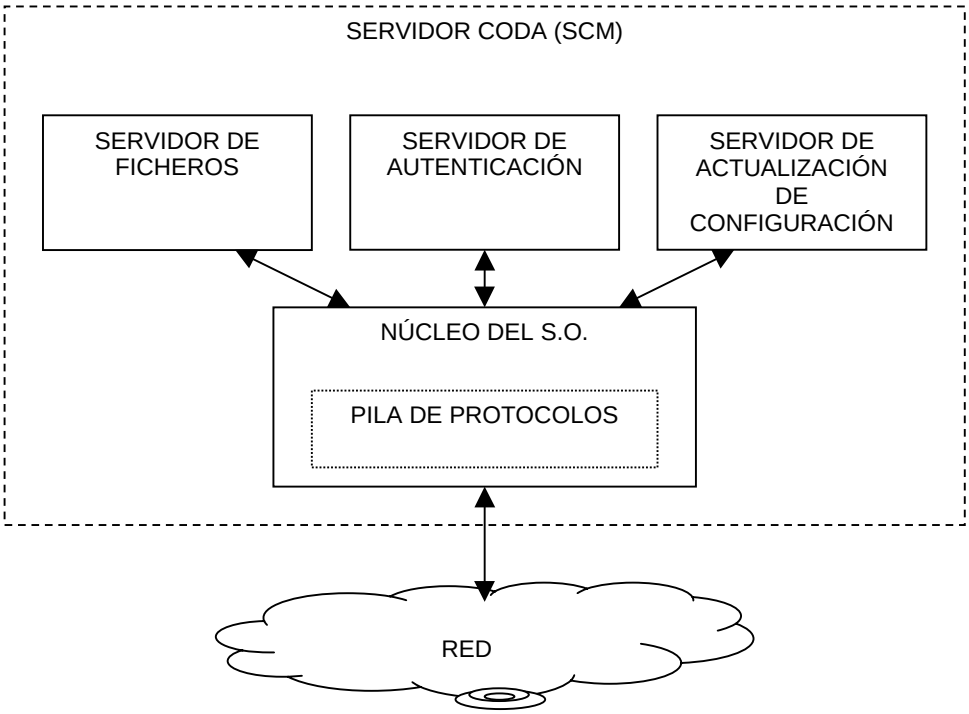


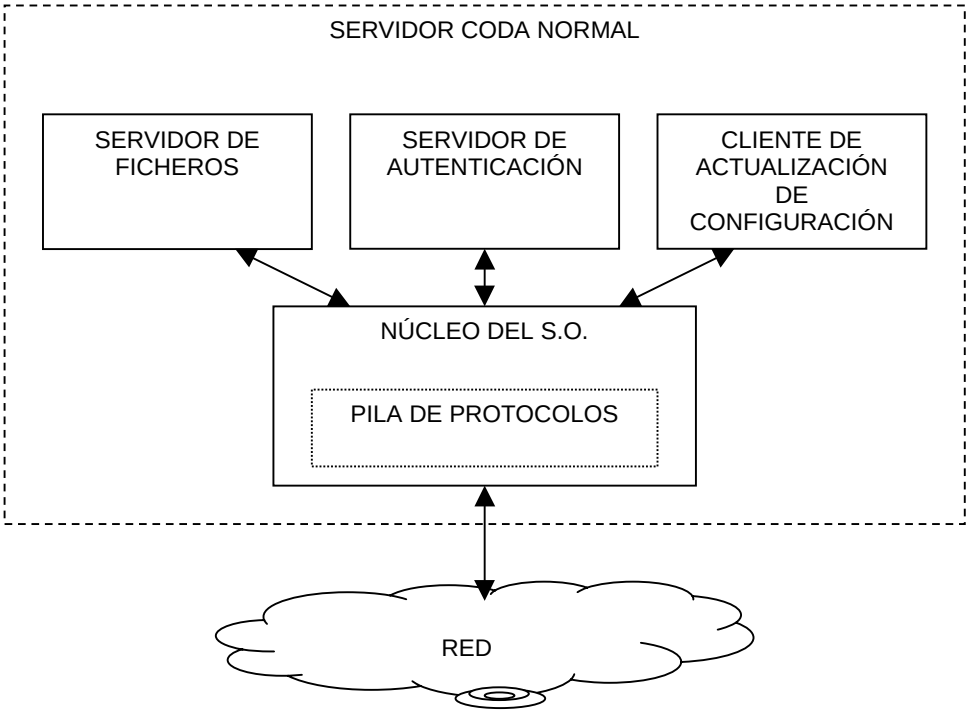
Cuando las aplicaciones realizan llamadas al sistema sobre algún fichero (remoto) de Coda, el núcleo traspasa la petición a Venus, el cliente de cache de Coda. El cliente mantiene una cache local donde busca primero el fichero requerido sobre el que realizar la operación; si no puede encontrarlo entonces tendrá que realizar una petición a un servidor remoto. Si el cliente trabaja en modo *offline* o desconectado, entonces sólo

podrá operar sobre los ficheros almacenados en la cache local, propagando los cambios al resto del sistema cuando se reconecte al mismo. Los clientes necesitan por tanto un núcleo con soporte para Coda.

2.4 ESTRUCTURA DE LOS SERVIDORES CODA

Seguidamente se muestra la estructura de un servidor Coda; en el caso de que además sea SCM, y en el de que sea un servidor normal:





Como se aprecia, la única diferencia entre el SCM y un servidor Coda normal es que el primero dispone de un servidor de configuración (el SCM es el único que puede modificar las bases de datos de configuración) y el segundo como cliente (realiza consultas de configuración). En cualquier caso, un servidor Coda dispone de un sistema servidor de ficheros, y de un sistema servidor de autenticación, contra el que tienen que identificarse los usuarios que pretenden acceder a los datos de dicho servidor.

2.5 FUNCIONAMIENTO DEL SISTEMA

En primer lugar, el software correspondiente está instalado tanto en los servidores como en los clientes Coda, y los servidores están configurados correctamente y ejecutándose. A continuación, desde los clientes, los usuarios han de autenticarse en el sistema Coda, de lo contrario tienen la posibilidad de utilizar el sistema pero con los permisos de un usuario genérico (lectura y *lookup*).

Los permisos que proporciona Coda son: `r` (lectura), `l` (*lookup*, obtención de información), `i` (creación), `d` (borrado), `w` (escritura), `k` (bloqueo), `a` (administración). Dichos permisos pueden ser positivos (habilitados) o negativos (denegados).

Seguidamente, los usuarios (desde los clientes) pueden acceder con normalidad a los ficheros proporcionados por los servidores Coda, como si formaran parte del sistema de ficheros local. Los clientes Coda cachean localmente las peticiones más recientes realizadas por los usuarios, para incrementar el rendimiento, e incluso los usuarios pueden aconsejar a sus clientes sobre los ficheros de uso más frecuente (aunque éstos no hayan sido pedidos aún), lo que en Coda se denomina *hoarding*.

La cache local permite operar a los clientes Coda desconectados del resto del sistema: las operaciones solicitadas por las aplicaciones sobre los ficheros en la cache son guardadas por el cliente, de forma que cuando éste se reconecta al sistema propaga los cambios realizados localmente a los servidores que contengan los ficheros.

Por otra parte, para el caso de que dos o más clientes hayan realizado cambios locales sobre los mismos ficheros, o si un servidor ha estado desconectado del sistema, Coda proporciona un mecanismo (en algunos casos automático, en otros manual) de resolución de conflictos. Este mecanismo permite mantener – en teoría – la consistencia de los datos del sistema.

3 CONFIGURACIÓN DE UN SISTEMA CODA EN DEBIAN

3.1 PASOS PRELIMINARES

En este apartado se describen los pasos a realizar para instalar un sistema Coda completo sobre la distribución Debian de Linux.

Como ejemplo, supondremos un sistema formado por dos estaciones de trabajo funcionando con Debian como sistema operativo, y conectadas entre sí mediante una red local. Una de las máquinas actuará como cliente de Coda, y la otra realizará las labores de servidor de Coda (SCM), y cliente de Coda. En la primera máquina sólo hay que realizar las tareas para configurar el cliente, y en la segunda – además – las requeridas para configurar el servidor de Coda.

Inicialmente, para facilitar la instalación de los paquetes mediante la utilidad `apt-get` de Debian, modificaremos en la máquina correspondiente el fichero `/etc/apt/sources.list`:

1) Cambiamos las líneas en las que aparece una fuente de instalación “estable” por “inestable”, pues Coda tal vez necesite paquetes no disponibles en la versión estable de la distribución, como es un *kernel* con soporte para Coda. Por ejemplo, cambiar la línea:

   ```
   deb http://ftp.de.debian.org/debian/ stable main non-free contrib
   ```

   por:

   ```
   deb http://ftp.de.debian.org/debian/ unstable main non-free contrib
   ```

2) Añadir la fuente de los paquetes de Coda (ubicación original):

   ```
   deb http://www.coda.cs.cmu.edu/pub/coda/linux debian/binary-i386
   ```

3.2 CONFIGURACIÓN DEL CLIENTE CODA

1) Primero es necesario instalar un núcleo con soporte para Coda (si no disponemos ya de él). Para ello, hacemos (por ejemplo):

   ```
   # apt-get install kernel-image-2.2.19
   ```

   Seguidamente hay que ejecutar la aplicación `modconf` para instalar el módulo para el sistema de ficheros de Coda, así como módulos adicionales (por ejemplo el módulo correspondiente a la interfaz de red utilizada), y reiniciar el sistema.

2) A continuación se instalan los paquetes del cliente Coda:

   ```
   # apt-get install coda-client
   ```

Si el cliente se instala correctamente se lanzará automáticamente el programa de configuración básica que realiza las siguientes operaciones:

- Crea y modifica `/etc/coda/venus.conf` añadiendo la dirección del servidor Coda por defecto al que se conectará el cliente y el tamaño de la caché de *Venus* en kb.
- Añade al fichero `/etc/services` distintos puertos y protocolos utilizados por varios servicios de Coda.
- Crea el pseudo dispositivo `/dev/cfs0` utilizado para la comunicación entre el *kernel* y Venus. Si el programa no lo crea directamente, deberemos crearlo manualmente mediante `mknod /dev/cfs0 c 67 0` ó `cd /dev ; ./MAKEDEV cfs0`

3) Por último, sólo queda arrancar el cliente de cache Venus:

```
# venus &
```

Esta operación, además de arrancar el gestor de caché, monta automáticamente en el directorio `/coda` el volumen raíz del servidor Coda (SCM). Un *script* en `/etc/init.d/` arranca automáticamente Venus cada vez que se reinicia el sistema, por lo que esta operación solo es necesaria realizarla la 1ª vez. No obstante, si alguna vez es necesario limpiar la cache local y rearrancar el cliente:

```
# venus –init &
```

En `/var/log/coda/venus.log` y `/var/log/coda/venus.err` podemos comprobar si Venus arrancó correctamente y, en caso contrario, cuales fueron los errores.

4) Si se desea parar el cliente de coda:

```
# vutil shutdown
# umount /coda
```

Aunque no entraremos a discutirlos, en el archivo `/etc/coda/venus.conf` se pueden configurar otros parámetros más avanzados del cliente Coda. Por último decir que en `/var/lib/coda/` es donde se almacena la cache del cliente (`./cache/`), los metadatos de los archivos de la cache (`./LOG` y `./DATA`) y los archivos en conflicto junto con sus *change modification logs* (`./spool/*.tar` y `./spool/*.cml`)

3.3 CONFIGURACIÓN DEL SERVIDOR CODA

1) Instalamos los paquetes del servidor de Coda:

```
# apt-get install coda-server
```

2) Organización de disco del servidor:

Para un funcionamiento óptimo es recomendable crear dos particiones (*raw*, sin sistema de ficheros) para la RVM (*Recoverable Virtual Memory*), que es utilizada por Coda para manejar la información de volumen, directorios, atributos y listas de

control de accesos (ACL's). También se pueden emplear simples ficheros en lugar de particiones, que la RVM utilizará igualmente para almacenar los metadatos de Coda. En nuestro caso optaremos por dos ficheros, `/var/coda/LOG` y `/var/coda/DATA`. Adicionalmente se puede crear una partición para `/vice` (directorio donde se almacena la configuración, bases de datos y *logs* del servidor Coda) para reducir problemas de disco (errores de disco, partición *root* pequeña) aunque nosotros no lo haremos. Por último, como convenio en Coda se utiliza el directorio `/vicepa` para almacenar los archivos compartidos. Se puede crear una partición para él o, en nuestro caso, crear un enlace a otra partición con suficiente espacio de disco para los ficheros que se espera servir. Por tanto, las operaciones a realizar en este punto son:

```
# touch /var/coda/LOG ; touch /var/coda/DATA
# mkdir /usr/coda/vicepa ; ln –s /usr/coda/vicepa /vicepa
```

3) A la hora de crear usuarios de Coda es necesario que previamente sean usuarios Unix. Por ello, antes de configurar el servidor debemos preparar un usuario Unix que será el administrador del sistema Coda. Podemos crear uno nuevo con `adduser –u uid` o anotar el *login* y *uid* de un usuario ya existente en `/etc/passwd`.

4) Ejecutamos el programa `vice-setup`, que permite configurar el servidor de forma interactiva. A continuación se resumen las operaciones que realiza este *script*:

   - Crea fichero `/etc/coda/server.conf`, donde irá escribiendo los distintos parámetros de configuración, como por ejemplo la ubicación del directorio raíz de configuración del servidor Coda (`/vice`) o la de los archivos o particiones de la RVM, etc.

   - Crea los testigos o *tokens* para comunicación segura entre servidores de una misma célula. El *script* solicita la introducción de tres *tokens*: el de el servidor de autenticación, el de la utilidad `volutil` de creación de volúmenes y el de los procesos de propagación y actualización de cambios del sistema. Estos *tokens* son almacenados en `/vice/db/*.tk`.

   - Tras configurar este servidor como SCM, crea el archivo `/vice/db/servers` con el nombre del *host* (lo toma directamente del archivo `/etc/hostname`) y el identificador de servidor (`serverid`) que introducimos por teclado. Esto es necesario para poder identificar los distintos servidores que componen una célula. El `serverid` debe estar comprendido entre 1-126 ó 128-245.

   - Crea el archivo `/vice/db/ROOTVOLUME` con el nombre que escojamos para el volumen raíz. Posteriormente crearemos el volumen raíz del SCM con el mismo nombre que hemos introducimos aquí.

   - Creación del superusuario o administrador de Coda. El *uid* y *username* que introduzcamos deberá coincidir con el del usuario Unix que hayamos designado previamente a tal efecto. El *password* por defecto del superusuario es `changeme`.

   - A continuación se pasa a configurar la RVM. En nuestro caso introduciremos la ruta de los ficheros (`/var/coda/LOG` y `/var/coda/DATA`) creados para las áreas

de *log* y datos de la RVM (en caso de tener particiones se pondría el nombre de dichos dispositivos). La elección de los tamaños de estas particiones o ficheros deberá escogerse en función del tamaño del área de almacenamiento de los archivos compartidos. Para nuestras necesidades nos basta con 2M para el *log* y 22M para los datos. El *script* procederá a inicializar los ficheros o particiones a los tamaños indicados.

- Por último, el *script* termina de configurar el servidor añadiendo diferentes archivos y directorios a `/vice`. Por ejemplo, el archivo `/vice/db/files` contiene los ficheros que los procesos de actualización deben mantener consistentes en todos los servidores. También permite configurar la ubicación de los archivos compartidos (nosotros seguimos el convenio de ubicarlo en `/vicepa`) y el número de ficheros que albergará.

5) Ejecutamos los procesos para arrancar el servidor (no es necesario repetirlos cada vez que se arranca el sistema pues de eso ya se encarga el *script* `coda-server` situado en `/etc/init.d/`):

   `# auth2` (servidor de autenticación)
   `# rpc2portmap` (habilitar las llamadas a procedimiento remoto)
   `# updatesrv` (servidor de actualización y propagación de cambios)
   `# updateclnt –h <hostname>` (cliente actualización y propagación de cambios)
   `# startserver &` (servidor de ficheros)

   NOTA: El parámetro `hostname` de `updateclnt` debe ser el nombre del SCM.

   Para comprobar que el servidor de ficheros arrancó correctamente deberemos examinar el *log* del servidor `/vice/srv/SrvLog`. Si no fuera así, habría que realizar modificaciones sobre los ficheros de configuración para solventar los fallos de arranque. En nuestro caso tuvimos que editar el archivo de configuración del servidor `/etc/coda/server.conf` para introducir la dirección IP del SCM ya que el servidor de ficheros tenía problemas con la de *loopback*.

6) Para finalizar, crearemos el volumen raíz del SCM, que es el que montan automáticamente todos los clientes en `/coda` y sobre el que se crea la estructura de directorios (y se montan otros volúmenes) que ven los clientes Coda:

   ```
   # createvol_rep coda.root E0000100 /vicepa
   ```

   Este comando crea un volumen replicado llamado `coda.root` (este nombre debe ser igual al que introducimos en `vice-setup`), con el identificador E0000100 del grupo de servidores (VSG) que almacenarán (en `/vicepa`) este volumen replicado; en nuestro caso el grupo lo forma el único servidor que tenemos.

   Además de crear el volumen en sí, crea/actualiza las bases de datos de los volúmenes del sistema de ficheros: `/vice/db/VLDB` (localización de todos los volúmenes activos del sistema), `/vice/db/VRDB` (volúmenes replicados existentes) y `/vice/db/VSGDB` (conjunto de servidores que pertenecen a un VSG). Dado que creamos el volumen raíz es replicado, cualquier servidor perteneciente a esta célula podría replicar el volumen raíz sin más que añadir su nombre a la VSGDB.

7) Para parar el servidor:

```
# volutil shutdown
```

Por último decir que en `/vice` se pueden encontrar distintos *logs* de los procesos de autenticación (en `./auth2/`), actualización (en `./misc/`), etc. que pueden ser útiles a la hora de resolver problemas del servidor.

4 PRUEBAS

Para comprobar las funcionalidades de Coda se han realizado una serie de pruebas sobre el sistema cuya instalación se ha descrito en el punto anterior (un servidor y dos clientes, uno de ellos compartiendo la misma máquina con el servidor).

En primer lugar, se crearon varios usuarios adicionales del sistema Coda y grupos de éstos (los usuarios se crearon antes dentro de Linux). Para ello se utilizó la herramienta `pdbtool` cuyo uso será explicado en el apartado 5 Gestión del Sistema.

Una vez hecho esto el usuario puede autenticarse en el sistema desde su cliente y acceder a Coda con los permisos correspondientes. Se utiliza para ello la herramienta `clog`. Por ejemplo:

```
[daniel@m4]$ clog daniel
Username: daniel
Password:

[daniel@m4]$
```

La herramienta fundamental utilizada desde los clientes para operar con Coda es `cfs`. Esta herramienta permite, entre otras cosas, ver y establecer los permisos de cada usuario o grupo sobre un determinado directorio. Para poder comprobar el correcto funcionamiento de los permisos, utilizamos `cfs` para establecer distintos permisos a los grupos creados (véase el apartado 5 para los detalles en el uso de `cfs`).

4.1 OPERACIÓN NORMAL DEL SISTEMA

Con el sistema de ficheros de Coda se opera de la misma forma que con el sistema de ficheros local. Para acceder a los ficheros de Coda, en el cliente correspondiente hay que situarse en el directorio `/coda`.

Desde aquí pueden ejecutarse todos los comandos de Unix/Linux para gestión de ficheros y directorios. No obstante, si el usuario actual no tiene en Coda los permisos correspondientes (como, por ejemplo, el de borrado) la ejecución de la orden dará un error, comprobando de este modo el correcto funcionamiento de los permisos de Coda.

Se comprueba asimismo que mientras los dos clientes están conectados al sistema, los cambios efectuados por uno de ellos se ven automáticamente en el otro, confirmando el buen funcionamiento de Coda en este caso. Por ejemplo:

```
[daniel@m4 /coda]$ ls
./  ../
[daniel@m4 /coda]$ mkdir danieldir
[daniel@m4 /coda]$ ls
./  ../  danieldir/
```

```
[pera@m3 /coda]$ ls
./  ../

[pera@m3 /coda]$ ls
./  ../  danieldir/
```

4.2 DESCONEXIÓN DE LOS CLIENTES

También se han realizado pruebas para comprobar como funciona el sistema cuando uno o varios clientes se desconectan y operan *offline* sobre el sistema de ficheros de Coda, y verificar el mecanismo de resolución de conflictos.

Para que un cliente se desconecte del sistema se utiliza la orden:

```
# cfs disconnect
```

Para reconectarse al sistema:

```
# cfs reconnect
```

Para comprobar el estado de la conexión del cliente al sistema:

```
# cfs cs
```

En el caso de que se produzca un conflicto, tras la reconexión de los clientes, y sea necesaria su resolución manual, existe la herramienta interactiva `repair` (véase el apartado 5 para más detalles).

NOTA: durante una desconexión es imposible autenticarse en el sistema puesto que no hay acceso al servidor de autenticación. Por tanto, si durante una desconexión se desea tener los permisos habituales de un usuario o grupo es necesario estar autenticado antes de que se produzca la desconexión.

A. Desconexión de un cliente, sin conflicto:

En este caso uno de los clientes se desconecta del sistema, y estando desconectado modifica un archivo dentro de `/coda` (en su cache local) y crea un directorio. Previamente, estando conectado, Venus copió el archivo en la cache del cliente al haber accedido al fichero o simplemente al hacer un `ls`. Mientras tanto el otro cliente – conectado – no observa ningún cambio. Cuando el cliente se reconecta los cambios se reintegran correctamente en el sistema.

B. Desconexión de un cliente, con conflicto:

En este caso uno de los clientes se desconecta del sistema, y estando desconectado borra un fichero en `/coda`. El otro cliente – conectado – no observa estos cambios, y a su vez modifica el mismo fichero, y crea un fichero nuevo.

Cuando el primer cliente se reconecta, se detecta que el sistema no realiza una reintegración correcta de los cambios: en el cliente que se reconecta se produce un error, y es necesario cerrarlo y arrancarlo de nuevo, borrando su cache local, con la orden:

```
# venus –init &
```

De esta forma irremediablemente se pierden todos los cambios locales realizados en el cliente que se desconectó.

C. Desconexión de dos clientes:

En este caso, los dos clientes de nuestro sistema se desconectan.

Se detecta que, independientemente de las modificaciones locales realizadas en cada cliente, el sistema no reintegra bien los cambios: es decir, si en el mismo directorio (por ejemplo `/coda`) ambos clientes realizan modificaciones, aunque sean sobre ficheros diferentes, parece que estas modificaciones entran siempre en conflicto.

De hecho, el primero de los clientes que se reconecta propaga correctamente los cambios al resto del sistema. Sin embargo, cuando se reconecta el segundo cliente se produce un error en éste y sólo se aprecian las modificaciones realizadas localmente en este segundo cliente; el otro cliente no ve ningún cambio.

En este caso es necesario también reiniciar el segundo cliente en reconectar, por lo que los cambios realizados localmente en dicho cliente también se pierden.

4.3 CONCLUSIONES

De las pruebas realizadas se desprende el hecho de que el sistema funciona correctamente si los clientes se encuentran conectados al sistema (en este caso el sistema sería similar a NFS, por ejemplo).

Sin embargo, el sistema no maneja bien la desconexión de los clientes y la resolución de los posibles conflictos asociados, al menos en la versión utilizada. Tal vez esto sea debido a la forma que emplea Coda para almacenar la información relativa a los ficheros; parece que no maneja adecuadamente los *fid* (identificadores de ficheros) que guarda internamente. Por ejemplo, si dos clientes desconectados crean sendos ficheros en el mismo directorio lo hacen con el mismo fid (aunque sean ficheros distintos) y esto provoca un conflicto que Coda no puede resolver. Otra posibilidad es que el sistema trate la información relativa a directorios como si fueran archivos, y al reintegrar se encuentra que un archivo (directorio en realidad) ha modificado su contenido (se le han añadido ficheros) y, por tanto, interpreta que existe conflicto.

En cualquier caso, parece que la implementación de esta versión de Coda no es correcta o al menos en Debian no funciona correctamente (puesto que no hemos encontrado errores de configuración ni en los servidores ni en los clientes). Por ejemplo, cuando existe un conflicto dentro de un directorio, el directorio debería convertirse en un enlace simbólico (para seguidamente utilizar la herramienta `repair`) y esto no sucede a pesar de que el sistema detecta el conflicto y solicita su reparación.

5 GESTIÓN DEL SISTEMA

5.1 GESTIÓN DE USUARIOS

La herramienta `pdbtool` permite, desde un servidor, crear/eliminar usuarios y grupos de Coda de forma sencilla:

```
# pdbtool
> nui username uid (crear nuevo usuario)
> ng groupname ownerid (crear nuevo grupo)
> ci user/group newid (cambiar identificador de usuario o grupo)
> ag groupname/id user/group (añadir usuario o grupo existentes a otro grupo)
> rmu username (elimina usuario)
> rmg groupname (elimina grupo)
> n user/group (lista información de usuario o grupo)
```

Hay que recordar que es necesario que el usuario Coda ya exista como usuario Unix, debiendo coincidir tanto el *login* como el *uid*. Tras crear un usuario con la herramienta anterior, es necesario registrarlo en las bases de datos del SCM, estableciéndole un *password*, mediante:

```
# au –h <nombre_servidor_scm> nu
```

que solicitará la autenticación como superusuario Coda para poder realizar la operación.

El nuevo usuario Coda puede ahora autentificarse desde un cliente usando `clog`, cambiar su *password* con `cpasswd –h <nombre_servidor_scm>` o hacer *logout* con `cunlog`.

5.2 CONTROL DE PERMISOS

Coda utiliza listas de control de accesos (ACL's) para establecer los permisos que tiene cada usuario o grupo sobre los directorios. Establecer permisos sobre archivos individuales solo es posible con `chmod`. Los permisos son `r`(lectura), `l`(visualizar), `i`(creación), `d`(borrar), `w`(sobreescribir), `k`(*lock*) y `a`(administración). La utilidad `cfs` permite, desde un cliente Coda, visualizar y establecer los permisos de cada directorio:

```
# cfs la directorio                  (lista los permisos sobre el directorio)
      System:AnyUser rl               (grupo especial con todos los usuarios)
      System:Administrators rlidwka   (grupo especial del administrador)
      User01 rl                       (usuario con permisos de lectura y
                                      visualización)

# cfs sa directorio grupo/usuario permisos      (establece permisos)
```

Para poder establecer permisos es necesario estar autentificado y tener o pertenecer a un grupo con el permiso de administración (`a`). Los grupos `System:AnyUser` y

`System:Administrators` se crean por defecto y el dueño de ambos es el superusuario de Coda.

5.3 DETECCIÓN DE PROBLEMAS

Cuando un servidor o cliente (Venus) se cae, no muere, ni vuelca un `core` como hacen la mayoría de los programas Unix, sino que se queda indefinidamente dormido para poder depurarlo con `gdb`, pero esto solo es interesante para los desarrolladores. La labor de los administradores es detectar el mal funcionamiento lo antes posible (examinado los *logs* del sistema `/vice/srv/SrvLog` para el servidor y `/var/log/coda/venus.log` para el cliente) y rearrancar el sistema, matando previamente los procesos dormidos.

Para descubrir la causa de los problemas del servidor, podemos reiniciarle con `startserver -d 10` para que vuelque mas información sobre su *log*. También sería interesante revisar los *logs* de los procesos *update*, *rpc* o autenticación. En el caso de Venus lo más útil es ejecutar la herramienta `codacon` en un terminal a parte para ver las interacciones de Venus con los servidores.

5.4 REPARACIÓN DE CONFLICTOS

Hay ocasiones en las que hay que resolver manualmente los conflictos cuando Coda no puede solventarlos directamente. Para ello Coda provee al administrador (o a un usuario con los permisos suficientes) la herramienta `repair` que facilita la tarea de reparación. Existen dos tipos de conflictos que se tratan de manera distinta:

a) *Server/server conflicts*, que básicamente ocurren cuando tras una desconexión nos encontramos con dos versiones distintas de un mismo archivo almacenadas en dos servidores. `repair` permite comparar las dos versiones existentes en cada servidor (con `compare`) y eliminar una y reintegrar la otra en todos los servidores (con `do`).

b) *Local/global conflicts*, que ocurren cuando un cliente desconectado modifica un archivo y durante esa desconexión otro cliente ha modificado ese mismo archivo en el servidor. `repair` permite ver las modificaciones realizadas por el cliente (con `listlocal`), las modificaciones pendientes (con `checklocal`), descartar una modificación del cliente (con `discardlocal`), llevar a cabo una modificación del cliente sobre el servidor (con `preservelocal`), etc.

5.5 GESTIÓN DE VOLÚMENES Y BACKUPS

De forma muy resumida: la herramienta `volutil` permite crear, borrar (`volutil purge`) y manipular volúmenes (realización de *backups*, etc). A la hora de hacer un *backup* de un volumen de un servidor hay que tener en cuenta que este puede estar siendo actualizado en cualquier momento. Por ello, es necesario crear una imagen de solo lectura del volumen que no contenga ninguna transacción o mutación incompleta (`volutil clone`). Seguidamente se debe volcar el volumen a un fichero o ficheros (`volutil dump`) para que posteriormente pueda ser copiados en los dispositivos de

almacenaje usuales (`volutil dump`). El volcado puede ser incremental, de modo que solo aquellos archivos que han sido modificados desde el último *backup* sean volcados y copiados. Por último, tras restaurar un volumen (con `volutil restore`) es necesario actualizar las bases de datos de volúmenes en los servidores Coda.

6 BIBLIOGRAFÍA

Para más información se puede consultar la *web* de Coda:

http://www.coda.cs.cmu.edu/doc/html/index.html

Allí están disponibles varios documentos que describen el funcionamiento y la instalación del sistema. Los principales son:

The Coda Administration and User Manual, en formato PostScript o en HTML.

The Coda HOWTO, en formato PostScript o en HTML.