Redimensionando discos y migrando a XFS

SeaJob
Sindominio.net

seajob@sindominio.net

A lo largo de este artículo, explico cómo he migrado mis particiones `/boot`, `/var`, `/usr` y `/home` a XFS, además de sacar 150 Mb de `/boot` y dárselos a `/var`.

Para ello, usaré tanto GNU/parted, como el paquete debian kernel-patch-xfs(o una versión del parche descargada de sgi), y el paquete xfsprogs.

Espero que os sirva de ayuda, pero no quiere ser ni siquiera un HOWTO;es simplemente una explicación de cómo lo he hecho yo. Algo muy parecido os debería funcionar, pero depende del estado actual de vuestro sistema.

Podeis encontrar este documento en otros formatos en mi página web (http://www.sindominio.net/~seajob/articulos/migrando_a_xfs/).

1. Situación de partida

Tengo un sistema GNU/Linux, corriendo Debian Sid. Tengo ya compilado y funcional un kernel 2.4.16 parcheado con soporte para XFS. Para parchear el nucleo, he usado el paquete kernel-patch-xfs. Más información podeis conseguir en LinuxDoc (http://www.linuxdoc.org), tanto en el kernel-howto como en el Linux + XFS HOWTO.

La situación de las particiones de mi disco era la siguiente antes de empezar:

Partición	Punto de montaje	Tamaño	Sistema de ficheros
/dev/hda2	/pruebas	2838 Mb	XFS
/dev/hda6	/boot	250 Mb	ext2
/dev/hda7	/var	250 Mb	ext2
/dev/hda8	/usr	3752 Mb	ext2
/dev/hda9	/home	6161 Mb	ext2

2. Redimensionando `/boot` y `/var`

2.1. Haciendo un backup de la partición

Puesto que voy a jugar con la tabla de particiones, lo mejor es hacer un backup de los datos. Tanto en /var como en /boot hay datos fundamentales para el normal funcionamiento del sistema, así que nos debemos asegurar de que no perdemos nada.

El método que he utilizado, siguiendo la recomendación del XML+Linux-HOWTO, es:

```
[root@supra/]$ telinit 1
[root@supra/]$ mkdir /otro-boot
[root@supra/]$ cd /boot
[root@supra/]$ tar lcf - .|(cd /otro-boot; tar xpvf -)
```

y algo similar para `/var`, con la diferencia de que éste lo vuelco en `/pruebas`.

Seguidamente, varío [1] `/etc/fstab` de forma que ya no monte `/boot`, y que monte en `/var /dev/hda2` (que es lo que está montado en `/pruebas`). Desmonto `/boot`, elimino el directorio, y renombre `/otro-boot` como `/boot`. Por si acaso, ejecuto **lilo**, y reinicio de nuevo en modo monousuario.

2.2. Usando GNU/parted

Ahora empiezo a usar parted, que me permite redimensionar las particiones. En principio, no deberían de perderse datos, pero por si acaso tenemos el backup.

Ejecuto **parted**, y tras la información sobre la GPL, obtengo un prompt [2] , y con el comando **print** obtengo la situación actual de la tabla de particiones:

```
(parted) print
Disk geometry for /dev/hda: 0.000-19077,187 megabytes
Disk label type: msdos
Minor    Start       End     Type      Filesystem  Flags
1          0,031   5004,624  primary   FAT         boot
2       5004,624   7867,771  primary   xfs
3       7867,771   8056,032  primary   ext2
4       8056,033  19077,187  extended
5       8056,063   8534,531  logical   linux-swap
6       8534,562   8753,285  logical   ext2
7       8753,315   9005      logical   ext2
8       9005,216  12817,485  logical   ext2
9      12817,516  19077,187  logical   ext2
(parted)
```

El primer objetivo es reducir unos 150 Mb hda6 (/boot), y pasárselo a hda7 (/var). [3]

Lo primero que debemos hacer es reducir hda6. La sintaxis es:

```
resize n_particion Mb_inicial Mb_final
```

donde n_particion es el número de la partición (el número en la columna "Minor" en la salida del **print**), Mb_inicial es el principio de la partición (expresado en Mb, aunque sea un poco chocante; nos valen como guía los valores en la columna "Start"), y Mb_final es el final de la partición (en Mb, la columna "End").

Con estos datos, el comando que ejecuto es:

```
(parted) resize 6 8534 8600
```

Ahora debemos seguir con la redimensión de `hda7`; existe un problema: **parted** no es capaz de mover el punto de inicio de una partición ext2, así que tendremos que hacer un poco de trampa: la eliminamos, y luego creamos una que ocupe todo el espacio vacío: [4]

```
(parted) rm 7
(parted) mkpartfs logical ext2 8600 9005
```

Hay que andar con mucho cuidado con lo que acabo de hacer!!!. He eliminado la partición 7, con lo que las particiones 8 y 9 pasan a ser la 7 y la 8, y la nueva que creemos será la 9. Si no reflejamos estos cambios en /etc/fstab antes de reiniciar, tendremos que montar a mano las particiones... Yo no me dí cuenta y tuve que hacerlo...

Ahora nos queda revisar que los cambios hayan funcionado. Bien con **fdisk**, bien con **parted**, o bien montandolos en `/mnt`, revisamos que las particiones estén integras y con sus nuevos tamaños.

3. Migrando las particiones a XFS

La parte más complicada (y, sobre todo, la más delicada) del proceso ya está hecha; a partir de ahora, lo único que me queda por hacer es ir haciendo backups de las particiones (las que aún no he hecho) e ir migrandolas de ext2 a XFS.

3.1. Migrando `/dev/hda6` y `/dev/hda9`

Este es el caso más sencillo. Tenemos 2 particiones en las que no tenemos datos de interés (puesto que los hemos movido antes), por lo que no tenemos que hacer mayores cambios.

El comando para migrar una partición a XFS es sencillo [5] :

```
supra:~# mkfs.xfs -f /dev/hda6
meta-data=/dev/hda6              isize=256    agcount=8, agsize=121993 blks
data     =                  s      bsize=4096   blocks=975940, imaxpct=25
         =                         sunit=0      swidth=0 blks, unwritten=0
naming   =version 2               bsize=4096
        log      =internal log           bsize=4096   blocks=1200
realtime =none                    extsz=65536  blocks=0, rtextents=0
supra:~#
```

Hacemos lo mismo para `/dev/hda9`, y ya tenemos 2 particiones (más la que teníamos al empezar) en XFS.

Ahora tengo que devolver los datos que corresponden a cada partición, para lo cual hago lo mismo que hice al principio para guardarlos:

```
[root@supra/]$ telinit 1
[root@supra/]$ mount -t xfs /dev/hda6 /mnt
[root@supra/]$ cd /boot
[root@supra/]$ tar lcf - .|(cd /mnt; tar xpvf -)
```

con lo que ya tengo los datos de `/boot` de vuelta en la partición original, y esta reducida.

Me aseguro de que efectivamente los datos están en `/mnt`, varío de nuevo `/etc/fstab` para que monte `/dev/hda6` en `/boot` al arrancar, dejando una linea como esta:

```
/dev/hda6       /boot     xfs      rw
```

Renombro el directorio `/boot` a `/boot-viejo`, y creo de nuevo `/boot`. Monto el nuevo. Por si acaso, ejecuto **/sbin/lilo** de nuevo [6].

Hago otro tanto para `/dev/hda9` y `/var`.

3.2. Migrando `/usr` y `/home`

Estos 2 sistemas de ficheros no suponen ninguna dificultad especial, puesto que no hay que redimensionarlos. Lo único que tengo que hacer es mover los datos a otra partición, mover el punto de montaje a esa, darle formato a la original y devolver a su sitio los datos. Además, cuento con una partición vacía, y que ya está formateada con XFS.

1. Los datos que tenía en `/home` no me entraban en la partición libre. Tampoco fue mucho problema, puesto que en `/usr` sí entraban. Así que moví datos de `/usr` al espacio libre, luego los de `/home` a `/usr`, desmonté `/home`, lo formateé, lo volví a montar, repuse los datos, y luego hice lo mismo con `/usr`.
2. Como juego con `/usr`, es posible que en algún momento me quede sin la gran mayoría de las aplicaciones (**vi** o **GNU/Emacs**). De hecho, me pasó :(Esto no es necesario, pero en el caso de que suceda con cambiar un par de nombres y hacer un montaje a mano todo debería volver a estar accesible..

3.3. Migrando `/`

Esto aún no lo he hecho, pero tampoco debería ser muy dificil: sólo tengo que mover todos los datos a la partción libre como ya he contado, cambiar `/etc/fstab`, reiniciar, formatear, recuperar los datos a la partición, volver a cambiar `/etc/fstab` y volver a reiniciar...

Supongo que en cuanto tenga un rato lo haré...

Notas

1. Antes de tocar `/etc/fstab`, es importante verificar que efectivamente los archivos están donde los hemos querido poner.
2. El listado es aproximado, puesto que lo he sacado despues de hacer los cambios...
3. **Creo** que sólo lo puedo hacer porque son particiones contiguas, aunque si no lo fueran podría hacer algunas cosas... pero sería bastante más largo.
4. Es posible que entre el rm y el mkpartfs tengamos que reiniciar el equipo, en el caso de que parted se queje de que la partición está en uso. Al menos en mi caso, ha sido la única forma de hacerlo.
5. De nuevo, los datos no son exactos.
6. Yo tomé una medida extra de precaución: cree una entrada nueva en `/etc/lilo.conf` apuntando a una imagen en `/boot-viejo` antes de ejecutar **/sbin/lilo**. No hizo falta, y de hecho no estoy seguro de que hubiese funcionado, pero no está de más...