JFS Comparative Test

ACCOUNTING FOR THE HARD DISK

A journaling file system is essential if Linux is to break into the enterprise market. At the moment there are four highly promising approaches, all at various stages of development, from virtually non-existent to ready to go. Bernard Kuhn delves deeper

Linux is rock solid in terms of workstation and server functionality. But those of us who simply have to have the latest red-hot kernel patches and hardware drivers, or who are simply involved in kernel development, will be no stranger to system crashes. And of course, not even the best system can keep going if there is a power cut (unless it has a highly expensive UPS system!).

No matter what the circumstances are that force Linux to its knees, after rebooting the rule is to do a hard disk check first of all. This inspects all the files and rarely completes inside ten minutes. Depending on the size of the file system and number of hard disks, the procedure may even take several hours. Worse still, in rare cases a manual intervention may even be necessary (fsck). Although it is unlikely, the data SNAFU will have played itself out completely if the file system can no longer be repaired. At this point the only thing that will help is to restore a hopefully up-to-date backup.

But this makes things sound worse than they really are. The Extended2-Filesystem has provided sterling service since 1993 for countless Linux servers, whose rare unplanned downtimes put the potential problems into perspective. However, Linux beginners and pros are longing for a file system which is completely ready to run after a nasty crash, without human assistance and within a few seconds. The magic word for the solution to this problem is journaling.

Table 1: File systems with journaling at a glance

Name	B-Tree	64-Bit clea	Development stage	Licence
ReiserFS	Yes	No	Ready for everyday use with restrictions	GPL
ext3	No	No	Fully-functioning Alpha test version	GPL
jfs (IBM)	Yes	Yes	Alpha test version	GPL
xfs (SGI)	Yes	Yes	Beta test version for kernel 2.4-series	GPL

Journaling

The "ordinary" ext2-filesystem sets a flag on sign-on (mount). This flag is only cancelled on an orderly sign-off (unmount). So after a crash the operating system can tell whether the disk has been cleanly unmounted or not: in other words, whether there is potentially inconsistent data on the disk.

In order to correct this fault all files must be checked individually, which can be a very tedious procedure (called *Recovery*). A solution to the problem is to record in a journal which files are being processed at any moment. Then, after a power cut, only the

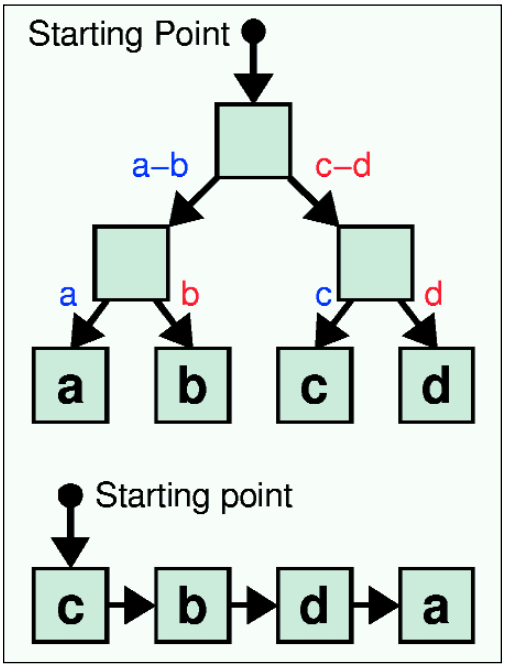

Fig.. 1: unbalanced (below) vs. full balanced tree (top)

files that were open at the time need be checked. In modern file systems a transaction-oriented approach is used, more often than not as long as any procedure has not been completely executed, the old data from the previous transaction retains its validity. This is especially important if for example a write process has had an unplanned interruption.

Balanced trees

Apart from brief recovery times, modern file systems are characterised by greater accessibility. This is achieved by using so-called *B-Trees* instead of the usual linear arrangement of data blocks. So, for example, in the ext2-filesystem directory entries are made in a linked list (see Fig. 1). If a directory has e.g. 1,000 entries, then on average some 500 search steps are necessary to find a file, while in an ideal balanced tree (binary tree) after just ten steps (ld 1000) the result is brought to light (compare Fig. 1 with four entries). The improvement in performance is, however, obtained at the expense of a considerably more complex (and thus error-prone) program code. In particular, after each new entry the tree has to be "re-balanced", so that all paths from the root to the most distant leaves remain roughly the same length. Seen like this, linked lists are completely degenerate balanced trees.

Practice

So much for dull theory. The complexity of B-Tree and journaling algorithms have so far made conversion into Linux reality difficult. Apart from the ready-

Recipe 1: ext3fs-retrofitting

Fitting an existing ext2-file system with journaling capabilities is, thanks to the backwards-compatible ext3-filesystem, almost childsplay for an advanced Linux user. Linux beginners have only to overcome the hurdle of the kernel compilation and installation. Obviously it is essential to back up all important files before carrying out this step, which does have its risks.

1. Firstly, you will need an unmodified kernel and the ext3 patch.

```
cd /tmp
wget ftp://ftp.uk.kernel.org/pub/linux/kernel/v2.2/linux-2.2
.13.tar.gz
wget ftp://ftp.uk.linux.org/pub/linux/sct/fs/jfs/old/ext3-0.
0.2c.tar.gz

There already exists a ext3-0.0.2f version, but this
one only applies to a prepatched Red**Hat kernel (2.2.16-3).
```

2. Now the kernel has to be unpacked, patched, configured and installed. Don't forget: during kernel configuration in the section File systems the option Second extended fs development code must be activated for ext3. After installing the kernel you should first ensure by doing a reboot that the system still starts up as usual.

```
cd /usr/src
rm linux # delete old link
tar -xzf /tmp/linux-2.2.13.tar.gz
tar -xzf /tmp/ext3-0.0.2c.tar.gz
cd linux
patch -p1 <\<> ../ext3-0.0.2c/linux-2.2.13-ext3.diff
make menuconfig
make clean && make dep && make bzImage
make modules && make modules_install
# after /boot over write kernel and instal by LILO
```

3. Now all non-root-partitions can be converted to the ext3-filesystem. To do this the user must manually install and initialise a journal file on the partition. Depending on the activity the journal should have a capacity of about 10 to 30 MB. For the initialisation the inode number, which the journal on the partition represents, is needed. This number can be found using the command "ls" with the option "-i". In the following example /usr is a correctly mounted ext2-formatted partition (/dev/hda4).

```
# in /etc/fstab for the /usr-input, replace the
# file system identifier "ext2" by "ext3"
vi /etc/fstab

# prepare /usr unmount (otherwise "busy")
init 1

# install journal (30MB)
dd if=/development/zero of=/usr/journal.dat bs=1k count=30000

# determine inode number (here e.g. 666)
ls -i /usr/journal.dat
   666 /usr/journal.dat

# mount /usr as ext3-fs and initialise journal with
# calculated inode number
umount /usr
mount -t ext3 /dev/hda4 /usr -o journal=666
```

So far, so good. Unfortunately the above method cannot be used on the root partition, since this cannot be unmounted during operation. The chicken-and-egg problem can be solved by performing the journal initialisation as a kernel boot option. So to do everything in sequence:

4. As with the above example, the computer has to be informed in /etc/fstab for future system starts that the root file system will henceforth be an ext3-filesystem (replace ext2 in the "/"-entry by ext3).

5. The journal is (as above) installed by hand on the root partition and the inode number (in this case, 7777) of the journal must be assigned as a kernel parameter

```
dd if=/dev/zero of=/journal.dat bs=1k count=30000
ls -i /journal.dat
  7777 /journal.dat
reboot
```

The computer now starts up again. When the LILO prompt appears a couple of additional kernel options including the inode number of the journals must be added to the initialisation:

```
LILO: linux ext3 rw rootflags=journal=7777
```

The root partition will now be available after a hard reset within a few seconds recovery time (or at least, it should be). The whole procedure can also be cancelled by again replacing ext3 in /etc/fstab by ext2.

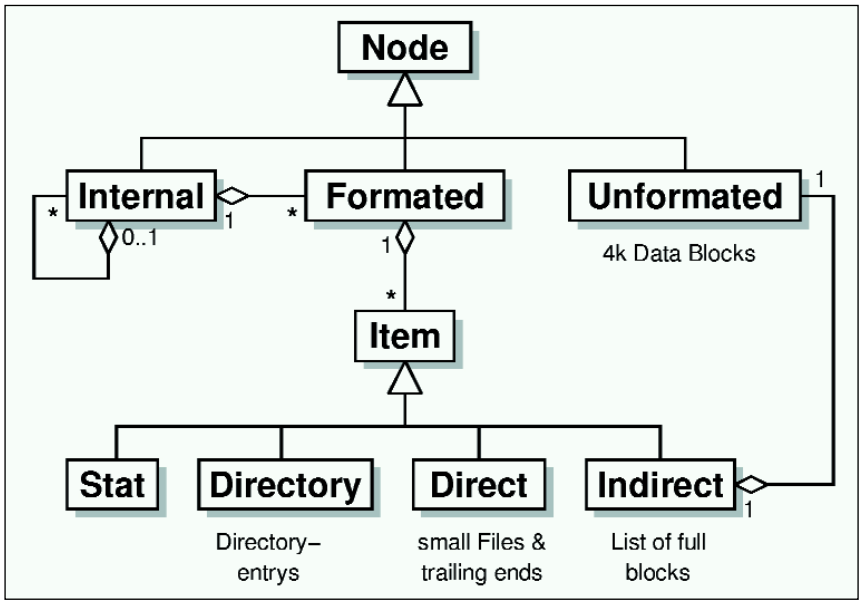

Fig. 2: The structure of the ReiserFS (simplified) in UML notation

made, free open-source ReiserFS, IBM and SGI are now rushing to ship their tried-and-tested and robust implementations JFS and XFS to Linux. But for anyone who was already satisfied with the ext2-file system and is only interested in short recovery times, a closer look at the ext3-fs will be worthwhile.

ext3-fs

The ext3-filesystem is merely an expansion of the well-known ext2-filesystem with journaling functionality and has no performance-boosting balanced trees. This means that existing Linux installations can continue to be used immediately on an ext2 base without reinstallation or time- and space-wasting copying actions, since ext3 is built on the basis of the existing structures [1]. On top of this, for advanced Linux users, installation and getting started are not especially complicated (see method 1). However, ext3fs is, according to the chief developer Stephen Tweedie, only in the alpha test phase and a long way from being suitable for everyday use. Nevertheless, there is a lot of positive feedback being gathered in news groups and other Internet forums. Also, a short test in our hardware lab did not find any weaknesses. But at the same time you must not forget that alpha test versions in Linux would be regarded by the 'marketing' department as the equivalent of Version 1.0 in many other operating systems.

ReiserFS

What began as a private study by the file system specialist Hans Reiser has now developed into a powerful file system which is suitable for everyday use [2]. Tests and experiments are however not yet completed and research is continuing into possible improvements - now at the request of SuSE GmbH.

The ReiserFS arranges files and directory entries into balanced trees. Small files or remnants of files (tail ends), directory entries and references to normal 4K file blocks (Unformatted Nodes) are all accommodated in 4K blocks (Formatted Nodes) in order to make best use of the available disk space (cf. Figure 2). A beneficial side effect of this arrangement is that you get more data in the buffer cache and therefore fewer disk accesses are necessary. With ReiserFS a watch is kept at all times to ensure that the data is kept close to its references and directory entries so that large movements of the write/read heads are avoided.

All these refinements have meant that the source code has grown five-fold compared with the ext2 file system. Nevertheless (or even because of this) there are currently still some restrictions imposed on ReiserFS: only 4k blocks are allowed and the use of SoftRAID is completely prohibited. Hardware platforms other than the x86 are also unsupported.

Unfortunately it is considerably more complicated to start up ReiserFS than it is with ext3 (see Recipe 2). As an alternative to time-consuming manual installation you may install Mandrake Linux 7.1 or SuSE Linux 6.4: both distributions offer ReiserFS as an alternative filesystem even on the level of the graphical installer.

After intensive tests by SuSE, some kernel developers consider ReiserFS is still not ready for mission critical use. In day-to-day work this file system has, however, already proven itself ideal for more than six months on the workstations of the author of this article. Daily backup of all important data on an NFS server (with tried and trusted ext2fs and a tape drive) is nevertheless vital in case of a full crash.

XFS

More than a year ago, SGI announced their "jewel in the crown" to be made available under GPL conditions for Linux. Unlike the other numerous and successful Open Source Projects from SGI, the XFS has got off to a sluggish start - the reasons for this being, among other things, that it just wasn't "open" for a while. SGI's programmers were in the process of removing foreign intellectual property from the source code and replaced it with their own re-implementations. First impression of the alpha test version proved that these radical measures didn't take down the robustness of the code with it. Currently, XFS for Linux is in Beta test stage and according to SGI, a production stable version for the kernel 2.4 series will be available soon [3].

JFS

IBM's Journaling File System for Linux was announced, surprisingly, at this year's Linux World Expo in New York. The currently available version (0.0.9) however is still at a very early stage of development. The robust, tried and tested source code for this is available as drop in replacement for the

Info

[1] Ext3-Download: ftp://ftp.uk.linux.org/pub/linux/sct/fs/jfs

[2] ReiserFS-Homepage: http://devlinux.com/projects/reiserfs

[3] XFS-Homepage: http://oss.sgi.com/projects/xfs

[4] JFS-Homepage: http://oss.software.ibm.com/developerworks/opensource/jfs/index.html

Method 2: ReiserFS conversion

Anyone wanting to convert their computer to ReiserFS has at present got their work cut out. Just as in the case of the ext3 retrofit, this procedure is not without hazards. However, since the existing system has to be copied across in the course of the retrofit, there is no need for a backup - provided no errors are made during repartitioning and there is a suitable boot diskette available in the case of a reconfigured LILO.

As part of the preparation a free partition is required, which has to be big enough to be able to accommodate the existing Linux installation (obviously the system can still consist of several partitions). In addition you will need an approx. 30 MB /boot partition (with ext2 file system), since LILO will not work with a kernel on a ReiserFS. /boot is mounted as read-only in normal operation, so that after an abrupt interruption to operations there is no need for fsck. But now, step by step:

1. *First the kernel sources and the patch for the journaling ReiserFS are needed. Warning: there is also a ReiserFS without journaling!*

```
cd /tmp
wget ftp://ftp.uk.kernel.org/pub/linux/kernel/v2.2/linux-2.2
.16.tar.gz
wget http://devlinux.com/pub/namesys/linux-2.2.16-reiserfs-3
.5.24-patch.gz
```

Unpack, patch, configure and install the kernel (Warning: Don't forget the option Filesystems/ReiserFS in configuration)

```
cd /usr/src
rm linux # delete old link
tar -xzf /tmp/linux-2.2.16.tar.gz
cd linux
gzip -cd /tmp/linux-2.2.16-reiserfs-3.5.24-patch.gz | patch -p1
make menuconfig
make clean && make dep && make bzImage
make modules && make modules_install
# copy kernel over after /boot and install via LILO
```

3. *After rebooting the tools (especially mkreiserfs) can now be prepared:*

```
cd /usr/src/linux/fs/reiserfs/utils
make
cp bin/reiserfs /sbin
```

4. *Setting up the new file systems and copying across data: in the following example /dev/hda2 is the current root partition (inc. /boot), /dev/hda6 is the future (journalled) root partition and /dev/hda5 the future /boot partition (ext2, r/o). The virgin journaling ReiserFS requires, after formatting, as much as 30 MB for the journal.*

```
# Set system to "back-up" mode
init 1

# back up root partition
mkdir /tmp/newroot
mkreiserfs /dev/hda6
mount /dev/hda6 /tmp/newroot
(cd && tar cplf - . –exclude boot) | (cd /tmp/newroot && tar x
pf -)

# copy over /boot
mkdir /tmp/newboot
mke2fs /deb/hda5
mount /dev/hda5 /tmp/newboot
(cd /boot && tar cpf - . ) | (cd /tmp/newboot && tar xpf -)
```

5. *Adapt fstab. Instead of ext2 for root, reiserfs must be substituted. Also, the root partition has now moved (hda2 after hda5). And don't forget the entry for the new /boot partition. So: instead of the old /etc/fstab entry for the above example*

```
/dev/hda2  /     ext2     defaults  1 1
```

the relevant part of the new /tmp/newroot/etc/fstab must look something like this:

```
/dev/hda5  /     reiserfs  defaults  1 1
/dev/hda6  /boot  ext2     ro       0 0
```

6. *The best way to check whether this comprehensive move has worked, risk-free, is with a boot diskette. This means the delicate Master Boot Record will be unaffected for now:*

```
# Create boot diskette
dd if=/usr/src/linux/arch/i386/boot/bzImage of=/dev/fd0
rdev /dev/fd0 /dev/hda6  # define new root partition
sync && reboot
```

Once the computer (hopefully) has booted up in the copied system, all that remains is to modify /etc/lilo.conf for the new environment. Before calling up LILO, however, the /boot partition has to be mounted writeable, since otherwise "lilo" will

```
mount -o remount,rw /boot
```

kernel source tree. Unfortunately the roughly 1.3 Megabyte tgz package [4] contains only sparse documentation. Nevertheless a glance at the source code reveals that the JFS also makes intensive use of balanced trees and appears to be 64bit-clean.

Conclusion

Four highly promising approaches for journalling raise great hopes that Linux will shortly be ascending into higher spheres. This feature is important, not only for enterprise servers, but also for the embedded Linux market, which is growing like wildfire. (In this application it is quite common for computers to be switched off without shutting down.) With XFS and JFS, two projects which have arisen out of commercial products have entered the race. Their existing and robust code is currently being brought up to scratch for Linux by the developers. But the easily-installed ext3 and in particular the ReiserFS are already there. The latter can even be choosen as alternative to ext2 within the graphical installers of the latest SuSE and Mandrake distributions (SuSE encourages their customers to do so). Although there are rumours that tell that ReiserFS isn't production stable, at least the author spent six month of daily work on ReiserFS-enhanced workstations – without any data loss! ■