SGI XFS on SuSE 7.1

CRASH PROOF

HARALD MILZ

Version 1.0 of the XFS journaling filesystem has been available for download on the SGI website since 1 May – including as a patch for the 2.4.2 kernel. The obvious thing would seem to be, therefore, to try building it into the SuSE Linux 7.1 kernel.

For long enough, UNIX users have been used to not being able to simply switch their systems off. However, failsafe systems which work with a log similar to databases, continuously recording any changes, have been available for a few years. The Reiser filesystem was introduced to the Linux world with SuSE 6.2, after Stephen Tweedie's Ext3 filesystem had already been available in a highly experimental and unstable form since the end of 1999.

SGI had already announced months ago that it would be porting XFS, known from the Irix environment, to Linux, and for the last few weeks it has been available as source code to everyone.

What makes XFS interesting are a number of features not previously available, or at least not in this combination:

- full 64-bit support
- quotas
- extended attributes, ACLs
- maximum file size 16TB on 4K pages and 64TB on 16K pages. If the block device layer has been converted to 64 bit, files up to a size of 9 exabytes (9 x 10^18 byte) are addressable.
- *xfsdump* and *xfsrestore* for filesystem back up. Usefully, dumps created on Irix can be restored on Linux and vice versa - despite different endianness.

- A data management API (DMAPI/XDSM) allows implementation of hierarchical storage management systems without any further kernel modifications.
- Using *xfs_growfs*, filesystems can grow while mounted (in fact, they have to be mounted to be able to grow). The number of inodes can be changed during operation.
- The log can be situated in a separate partition or a different logical volume. This will only improve throughput if the log is kept on a different physical disk.

Patching the kernel

In order to put XFS on their machine, the user must first build the two patches *linux-2.4-xfs-1.0 .patch.gz* and *linux-2.4.2-core-xfs-1.0 .patch.gz* into the kernel in the normal way. The SGI website recommends using a standard kernel from ftp.kernel.org. RPMs and an installer are available for Red Hat.

However, our point of interest is the comparison with Reiser FS, up to now the only log-based popular filesystem - and it therefore seems sensible to test it with the SuSE 2.4.2 kernel. What could be awkward is the fact that this kernel already contains a whole variety of patches which make further modifications impossible, or at least difficult.

No need to worry though - apart from one reject for one makefile, the core patch runs without any problems. The reject can be safely ignored; the corresponding patch is already contained in SuSE's 2.4.2.

The page buffer and XFS options must be activated as part of the kernel configuration, and possibly also DMAPI. All other XFS options are primarily for error detection and are not required in our case. However, the core patch does produce one stumbling block. The top makefile suddenly contains the line:

```
CC = $(CROSS_COMPILE)gcc \
     -V egcs-2.91.66
```

This call will fail unless you happen to have that version of egcs installed. SuSE 7.1 normally comes with release 2.95.2, so this line should be commented out. This is the most obvious example of Red Hat imitation.

Enormous XFS module

Finally, the kernel is converted as usual with

```
make dep bzimage modules modules_install
```

make modules_install creates a new directory */lib/modules/2.4.2-XFS* for the modules. This is also where the three new modules *pagebuf.o*, *xfs_support.o* and *xfs.o* are located.

After updating */etc/lilo.conf*, calling *lilo* and rebooting, the modules can be loaded:

Listing

Kernel messages during mount

```
page_buf cache Copyright (c) 2000 Silicon Graphics, Inc.
XFS filesystem Copyright (c) 2000 Silicon Graphics, Inc.
Start mounting filesystem: lvm(58,14)
Starting XFS recovery on filesystem: lvm(58,14) (dev: 58/14)
Ending XFS recovery on filesystem: lvm(58,14) (dev: 58/14)
```

Log space required

```
seneca:/mnt #  df /dev/vg01/xfstest
Filesystem         1k-blocks     Used Available Use% Mounted on
/dev/vg01/xfstest      650560    13752    636808   3% /mnt
seneca:/mnt # du -s -k .
13368   .
```

Newly created 64MB filesystem

```
Filesystem         1k-blocks     Used Available Use% Mounted on
/dev/vg00/testlv       65528    32840    32688  51% /mnt
/dev/vg01/xfstest      60736       80    60656   1% /mnt
```

```
xfs       403600   0  (unused)
xfs_support 8400   0  [xfs]
pagebuf    23040   0  [xfs]
```

As you can tell from the number of pages it occupies, the XFS module itself is pretty extensive.

In order to be able to actually install and use a filesystem, the tools in the *xfsprogs* package have to be translated and installed in */usr/local*. *e2fsprogs-devel* must be installed before you can run *configure*, and two lines must be commented out in *include/liblvm.h*:

```
/*
#include "lvm_log.h"
#include "lvm_config.h"
*/
```

If you like, you can build an RPM package from the *xfsprogs*.

Man page muddle

When trying to access the XFS man page you will discover an annoying similarity in names as the X fontserver man page appears. The XFS man page is actually called *man 5 xfs*.

Filesystem performance is most easily measured using the tried and tested filesystem benchmark Bonnie. This benchmark tests I/O-access speed with a large file. First of all it performs a character-oriented write, then it repeats the operation (rewrite) and finally it performs a block-oriented write. Reading is character-oriented to start with and then block reads. To finish off, there are random searches.

Bonnie results

```
          ——-Sequential Output——— —-Sequential Input— —Random—
          -Per Char- —Block—- -Rewrite— -Per Char- —Block—- —Seeks—
Machine   MB  K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU K/sec %CPU /sec  %CPU

XFS-IDE   512  2933 99.5 16210 28.7  5463 15.1  2755 89.1 15384 19.7 222.7  3.7
Reiser-I  512  2832 99.4 18695 65.1  5279 16.0  2783 88.9 16348 23.4 218.0  4.1
EXT2-IDE  512  2919 98.6 14652 26.7  4045 12.7  2643 83.3 10546 11.5 159.9  2.3

XFS-SCSI  512  2932 99.5 15241 28.7  5529 14.2  2795 90.8 15380 20.0 215.8  3.6
Reiser-S  512  2838 99.5 18533 61.2  5360 15.1  2733 87.6 15412 21.8 209.4  3.5
EXT2-SCS  512  2967 99.4 22915 38.8  5251 12.6  2670 84.2 15018 16.5 213.0  2.6
```

Bonnie should be included in most distributions, so anyone ought to be able to use it.

The results (see box "Bonnie results") relate to the following test environment: AMD K6-II/350 on Tyan S1590S board, VIA Apollo chip set with VT82C586B IDE controller. The SCSI controller was a Symbios Logic 53C875 with a SYM53C8XX driver. Hard disks: IBM DJNA-352030 (EIDE, UDMA-33), IBM DNES-309170W (Fast-20 Wide SCSI). In each case Bonnie was running with a 512MB test file, about double the size of the main memory, to exclude possible cache effects. The first three lines relate to the EIDE disk, the last three to the SCSI disk.

Reiser FS appears to perform better in terms of speed, particularly when writing large blocks, but at the price of a considerable CPU load. If an actual application is CPU intensive and wants to write large blocks, it is entirely possible that a system using XFS would provide a greater throughput. On the other hand, XFS seems to have a slight advantage in the random seeks - especially useful when running an OLTP database. There is hardly any difference between the other variables - the variations are within tolerance levels.

Lame duck on IDE: Ext2

A comparison with the venerable Ext2 on the same test partitions also produces significant results. While quite markedly lagging behind on the IDE disk, Ext2 does play one or two trumps on the SCSI disk, and is considerably faster when reading blocks, but not when writing them, contrary to what you might expect. It could not be determined why Ext2 was so much slower on the IDE disk, but the fact was confirmed through additional tests.

As a little endurance test, XFS was subjected to eight hours of furious copying activity with a multitude of small files. It survived without the slightest problem. Another interesting comparison is to see how long it takes to delete firstly lots of small files and secondly a really large one (filled from */dev/zero*). The purpose was primarily to see how quickly the filesystem deletes indirect and multiple indirect blocks. Reiser FS with its tree structure claims to be much faster than the bitmap-oriented Ext2. The result can be seen in the diagram.

Finally for the question of how failsafe XFS is. A simple test is a *findlcpio*, copying the kernel source tree into the XFS filesystem, and pressing reset halfway through the process. The subsequent mount shows that the filesystem takes less than one second for the repairs (see "Kernel messages during mount").

Space saving: XFS

Compared to Reiser FS, XFS uses considerably less space for its log. If you create the filesystem with the default values, the log is also situated on the block device, but it still requires far less space (Listing, section "Log space required"). With Reiser FS, even a newly created filesystem already permanently occupies about 32MB. The underlying logical volume has a size of exactly 64MB (see third section of Listing).

Another feature: *mkfs.xfs* realises if a logical volume already contains a formatted filesystem (not only XFS, Ext2 is also recognised), and requires the option *-f* to carry on formatting regardless.

Further endurance tests are needed to see whether XFS is suitable for everyday use - so far it would appear that XFS lives up to SGI's promises. At the time of writing there wasn't any news on whether SuSE is working on an official patch for SuSE 7.1 - requests for information regarding this question drew a blank. Possibly it will take until version 7.2 in the middle of this year before anything happens in this respect. ■

Info

XFS-Homepage: http://linux-xfs.sgi.com/projects/xfs/
Further comparison tests: http://slashdot.org/developers/01/05/10/1747213.shtml

With the loss of a whole file tree, XFS is slow compared to Reiser FS and Ext2. With large files it is much faster than the rivals.