The SGI XFS Filesystem

XFS is a high performance journaling filesystem which originated on the SGI IRIX platform. It is completely multi-threaded, can support large files and large filesystems, extended attributes, variable block sizes, is extent based, and makes extensive use of Btrees (directories, extents, free space) to aid both performance and scalability.

Refer to the documentation at http://oss.sgi.com/projects/xfs/ for further details. This implementation is on-disk compatible with the IRIX version of XFS.

Mount Options

When mounting an XFS filesystem, the following options are accepted.

allocsize=size
: Sets the buffered I/O end-of-file preallocation size when doing delayed allocation writeout (default size is 64KiB). Valid values for this option are page size (typically 4KiB) through to 1GiB, inclusive, in power-of-2 increments.

attr2/noattr2
: The options enable/disable (default is disabled for backward compatibility on-disk) an "opportunistic" improvement to be made in the way inline extended attributes are stored on-disk. When the new form is used for the first time (by setting or removing extended attributes) the on-disk superblock feature bit field will be updated to reflect this format being in use.

barrier
: Enables the use of block layer write barriers for writes into the journal and unwritten extent conversion. This allows for drive level write caching to be enabled, for devices that support write barriers.

dmapi
: Enable the DMAPI (Data Management API) event callouts. Use with the "mtpt" option.

grpid/bsdgroups and nogrpid/sysvgroups
: These options define what group ID a newly created file gets. When grpid is set, it takes the group ID of the directory in which it is created; otherwise (the default) it takes the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is a directory itself.

ihashsize=value
: In memory inode hashes have been removed, so this option has no function as of August 2007. Option is deprecated.

ikeep/noikeep
      When ikeep is specified, XFS does not delete empty inode clusters
      and keeps them around on disk. ikeep is the traditional XFS
      behaviour. When noikeep is specified, empty inode clusters
      are returned to the free space pool. The default is noikeep for
      non-DMAPI mounts, while ikeep is the default when DMAPI is in use.

inode64
      Indicates that XFS is allowed to create inodes at any location
      in the filesystem, including those which will result in inode
      numbers occupying more than 32 bits of significance. This is
      provided for backwards compatibility, but causes problems for
      backup applications that cannot handle large inode numbers.

largeio/nolargeio
      If "nolargeio" is specified, the optimal I/O reported in
      st_blksize by stat(2) will be as small as possible to allow user
      applications to avoid inefficient read/modify/write I/O.
      If "largeio" specified, a filesystem that has a "swidth" specified
      will return the "swidth" value (in bytes) in st_blksize. If the
      filesystem does not have a "swidth" specified but does specify
      an "allocsize" then "allocsize" (in bytes) will be returned
      instead.
      If neither of these two options are specified, then filesystem
      will behave as if "nolargeio" was specified.

logbufs=value
      Set the number of in-memory log buffers. Valid numbers range
      from 2-8 inclusive.
      The default value is 8 buffers for filesystems with a
      blocksize of 64KiB, 4 buffers for filesystems with a blocksize
      of 32KiB, 3 buffers for filesystems with a blocksize of 16KiB
      and 2 buffers for all other configurations. Increasing the
      number of buffers may increase performance on some workloads
      at the cost of the memory used for the additional log buffers
      and their associated control structures.

logbsize=value
      Set the size of each in-memory log buffer.
      Size may be specified in bytes, or in kilobytes with a "k" suffix.
      Valid sizes for version 1 and version 2 logs are 16384 (16k) and
      32768 (32k). Valid sizes for version 2 logs also include
      65536 (64k), 131072 (128k) and 262144 (256k).
      The default value for machines with more than 32MiB of memory
      is 32768, machines with less memory use 16384 by default.

logdev=device and rtdev=device
      Use an external log (metadata journal) and/or real-time device.
      An XFS filesystem has up to three parts: a data section, a log
      section, and a real-time section. The real-time section is
      optional, and the log section can be separate from the data
      section or contained within it.

mtpt=mountpoint
      Use with the "dmapi" option. The value specified here will be

included in the DMAPI mount event, and should be the path of
the actual mountpoint that is used.

noalign
Data allocations will not be aligned at stripe unit boundaries.

noatime
Access timestamps are not updated when a file is read.

norecovery
The filesystem will be mounted without running log recovery.
If the filesystem was not cleanly unmounted, it is likely to
be inconsistent when mounted in "norecovery" mode.
Some files or directories may not be accessible because of this.
Filesystems mounted "norecovery" must be mounted read-only or
the mount will fail.

nouuid
Don't check for double mounted file systems using the file system uuid.
This is useful to mount LVM snapshot volumes.

osyncisosync
Make O_SYNC writes implement true O_SYNC. WITHOUT this option,
Linux XFS behaves as if an "osyncisdsync" option is used,
which will make writes to files opened with the O_SYNC flag set
behave as if the O_DSYNC flag had been used instead.
This can result in better performance without compromising
data safety.
However if this option is not in effect, timestamp updates from
O_SYNC writes can be lost if the system crashes.
If timestamp updates are critical, use the osyncisosync option.

uquota/usrquota/uqnoenforce/quota
User disk quota accounting enabled, and limits (optionally)
enforced. Refer to xfs_quota(8) for further details.

gquota/grpquota/gqnoenforce
Group disk quota accounting enabled and limits (optionally)
enforced. Refer to xfs_quota(8) for further details.

pquota/prjquota/pqnoenforce
Project disk quota accounting enabled and limits (optionally)
enforced. Refer to xfs_quota(8) for further details.

sunit=value and swidth=value
Used to specify the stripe unit and width for a RAID device or
a stripe volume. "value" must be specified in 512-byte block
units.
If this option is not specified and the filesystem was made on
a stripe volume or the stripe width or unit were specified for
the RAID device at mkfs time, then the mount system call will
restore the value from the superblock. For filesystems that
are made directly on RAID devices, these options can be used
to override the information in the superblock if the underlying
disk layout changes after the filesystem has been created.
The "swidth" option is required if the "sunit" option has been

specified, and must be a multiple of the "sunit" value.

swalloc
Data allocations will be rounded up to stripe width boundaries when the current end of file is being extended and the file size is larger than the stripe width size.

sysctls

The following sysctls are available for the XFS filesystem:

fs.xfs.stats_clear (Min: 0 Default: 0 Max: 1)
Setting this to "1" clears accumulated XFS statistics in /proc/fs/xfs/stat. It then immediately resets to "0".

fs.xfs.xfssyncd_centisecs (Min: 100 Default: 3000 Max: 720000)
The interval at which the xfssyncd thread flushes metadata out to disk. This thread will flush log activity out, and do some processing on unlinked inodes.

fs.xfs.xfsbufd_centisecs (Min: 50 Default: 100 Max: 3000)
The interval at which xfsbufd scans the dirty metadata buffers list.

fs.xfs.age_buffer_centisecs (Min: 100 Default: 1500 Max: 720000)
The age at which xfsbufd flushes dirty metadata buffers to disk.

fs.xfs.error_level (Min: 0 Default: 3 Max: 11)
A volume knob for error reporting when internal errors occur. This will generate detailed messages & backtraces for filesystem shutdowns, for example. Current threshold values are:

XFS_ERRLEVEL_OFF:	0
XFS_ERRLEVEL_LOW:	1
XFS_ERRLEVEL_HIGH:	5

fs.xfs.panic_mask (Min: 0 Default: 0 Max: 127)
Causes certain error conditions to call BUG(). Value is a bitmask; AND together the tags which represent errors which should cause panics:

XFS_NO_PTAG	0
XFS_PTAG_IFLUSH	0x00000001
XFS_PTAG_LOGRES	0x00000002
XFS_PTAG_AILDELETE	0x00000004
XFS_PTAG_ERROR_REPORT	0x00000008
XFS_PTAG_SHUTDOWN_CORRUPT	0x00000010
XFS_PTAG_SHUTDOWN_IOERROR	0x00000020
XFS_PTAG_SHUTDOWN_LOGERROR	0x00000040

This option is intended for debugging only.

fs.xfs.irix_symlink_mode (Min: 0 Default: 0 Max: 1)
Controls whether symlinks are created with mode 0777 (default) or whether their mode is affected by the umask (irix mode).

fs.xfs.irix_sgid_inherit (Min: 0 Default: 0 Max: 1)
Controls files created in SGID directories.
If the group ID of the new file does not match the effective group ID or one of the supplementary group IDs of the parent dir, the ISGID bit is cleared if the irix_sgid_inherit compatibility sysctl is set.

fs.xfs.inherit_sync (Min: 0 Default: 1 Max: 1)
Setting this to "1" will cause the "sync" flag set by the xfs_io(8) chattr command on a directory to be inherited by files in that directory.

fs.xfs.inherit_nodump (Min: 0 Default: 1 Max: 1)
Setting this to "1" will cause the "nodump" flag set by the xfs_io(8) chattr command on a directory to be inherited by files in that directory.

fs.xfs.inherit_noatime (Min: 0 Default: 1 Max: 1)
Setting this to "1" will cause the "noatime" flag set by the xfs_io(8) chattr command on a directory to be inherited by files in that directory.

fs.xfs.inherit_nosymlinks (Min: 0 Default: 1 Max: 1)
Setting this to "1" will cause the "nosymlinks" flag set by the xfs_io(8) chattr command on a directory to be inherited by files in that directory.

fs.xfs.rotorstep (Min: 1 Default: 1 Max: 256)
In "inode32" allocation mode, this option determines how many files the allocator attempts to allocate in the same allocation group before moving to the next allocation group. The intent is to control the rate at which the allocator moves between allocation groups when allocating extents for new files.