XFS

XFS is a high-performance journaling file system created by Silicon Graphics, Inc. XFS is particularly proficient at parallel IO due to its allocation group based design. This enables extreme scalability of IO threads, filesystem bandwidth, file and filesystem size when spanning multiple storage devices.

Related articles

File systems

Contents

1 Preparation

For XFS userspace utilities **install** the **xfsprogs (https://archlinux.org/packages/?name=xfsprogs)** package. It contains the tools necessary to manage an XFS file system.

2 Creation

To create a new filesystem on *device* use:

```
# mkfs.xfs device
```

In general, the default options are optimal for common use.**[1] (https://xfs.org/index.php/XFS_FAQ#Q:_I_want_to_tune_my_XFS_filesystems_for_.3Csomething.3E)[2] (https://access.redhat.com/documentation/en_us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-xfs#xfscreating)**

Sample output:

```
meta-data=/dev/device            isize=256    agcount=4, agsize=3277258 blks
         =                       sectsz=512   attr=2
data     =                       bsize=4096   blocks=13109032, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=6400, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Tip:

- One can optionally assign a label to the filesystem by using the `-L label` option.
- When using *mkfs.xfs* on a block device containing an existing file system, add the `-f` option to overwrite that file system.**[3] (https://access.redhat.com/documentation/en_us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-xfs#xfscreating)**. ***This operation will destroy all data contained in the previous filesystem***.

Note: "After an XFS file system is created, its size cannot be reduced. However, it can still be enlarged using the xfs_growfs command."**[4] (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/ch-xfs#xfscreating)** See **#Resize**.

2.1 Checksumming

xfsprogs (https://archlinux.org/packages/?name=xfsprogs) 3.2.0 has introduced a new on-disk format (v5) that includes a metadata checksum scheme called **Self-Describing Metadata (https://docs.kernel.org/filesystems/xfs-self-describing-metadata.html)**. Based upon CRC32 it provides for example additional protection against metadata corruption during unexpected power losses. Checksum is enabled by default when using **xfsprogs (https://archlinux.org/packages/?name=xfsprogs)** 3.2.3 or later. If you need read-write mountable xfs for older kernel, it can be easily disabled using the `-m crc=0` switch when calling **mkfs.xfs(8) (https://man.archlinux.org/man/mkfs.xfs.8)**:

```
# mkfs.xfs -m crc=0 /dev/target_partition
```

Note: Disabling metadata CRCs will also have the effect of disabling support for the **#Free inode btree**, **#Reverse mapping btree** and **#Big timestamps** features below, as well as

"reference count btrees" (see **mkfs.xfs(8) § OPTIONS (https://man.archlinux.org/man/mkfs.xfs.8#OPTIONS)** for details).

The XFS v5 on-disk format is considered stable for production workloads starting in Linux Kernel 3.15.

Note: Unlike **Btrfs** and **ZFS**, the CRC32 checksum only applies to the metadata and not actual data.

2.2 Free inode btree

Starting in Linux 3.16, XFS has added a btree that tracks free inodes. It is equivalent to the existing inode allocation btree with the exception that the free inode btree tracks inode chunks with at least one free inode. The purpose is to improve lookups for free inode clusters for inode allocation. It improves performance on aged filesystems i.e. months or years down the track when you have added and removed millions of files to/from the filesystem. Using this feature does not impact overall filesystem reliability level or recovery capabilities.

This feature relies on the new v5 on-disk format that has been considered stable for production workloads starting Linux Kernel 3.15. It does not change existing on-disk structures, but adds a new one that must remain consistent with the inode allocation btree; for this reason older kernels will only be able to mount read-only filesystems with the free inode btree feature.

The feature is enabled by default when using xfsprogs 3.2.3 or later. If you need a writable filesystem for older kernels, it can be disable with the `finobt=0` switch when formatting an XFS partition. You will need `crc=0` together:

```
# mkfs.xfs -m crc=0,finobt=0 /dev/target_partition
```

or shortly (because `finobt` depends on `crc`):

```
# mkfs.xfs -m crc=0 /dev/target_partition
```

2.3 Reverse mapping btree

The reverse mapping btree is at its core “a secondary index of storage space usage that effectively provides a redundant copy of primary space usage metadata. This adds some overhead to filesystem operations, but its inclusion in a filesystem makes cross-referencing very fast. It is an essential feature for repairing filesystems online because we can rebuild damaged primary metadata from the secondary copy.”**[5] (https://blogs.oracle.com/linux/xfs-online-filesystem-checking)**

“The feature graduated from EXPERIMENTAL status in Linux 4.16 and is production ready. However, online filesystem checking and repair is (so far) the only use case for this feature, so it will remain opt-in at least until online checking graduates to production readiness.”

From **mkfs.xfs(8) § OPTIONS (https://man.archlinux.org/man/mkfs.xfs.8#OPTIONS)**:

"The reverse mapping btree maps filesystem blocks to the owner of the filesystem block. Most of the mappings will be to an inode number and an offset, though there will also be mappings to filesystem metadata. This secondary metadata can be used to validate the primary metadata or to pinpoint exactly which data has been lost when a disk error occurs."

See also [6] (https://kernelnewbies.org/Linux_4.16#XFS_reverse_mapping_and_reflink_features_are_now_stable) and [7] (https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=35a891be96f1f8e1227e6ad3ca827b8a08ce47ea) for more information.

To try out this feature or future-proof new filesystems, pass the `-m rmapbt=1` parameter during filesystem creation:

```
# mkfs.xfs -m rmapbt=1 device
```

2.4 Big timestamps

Starting in Linux 5.10, XFS supports using refactored "timestamp and inode encoding functions to handle timestamps as a 64-bit nanosecond counter and bit shifting to increase the effective size. This now allows XFS to run well past the **Year 2038 problem** to now the Year 2486. Making a new XFS file-system with *bigtime* enabled allows a timestamp range from December 1901 to July 2486 rather than December 1901 to January 2038." The feature will also allow quota timer expirations from January 1970 to July 2486 rather than January 1970 to February 2106.

Big timestamps are enabled by default for new filesystems as of xfsprogs 5.15.

2.4.1 Upgrading

Verify whether an existing filesystem has bigtime enabled with **xfs_info(8) (https://man.archlinux.org/man/xfs_info.8)**:

```
# xfs_info / | grep bigtime
... bigtime=0 ...
```

With **xfsprogs (https://archlinux.org/packages/?name=xfsprogs)** 5.11 and newer you can upgrade an existing (unmounted) filesystem with **xfs_admin(8) (https://man.archlinux.org/man/xfs_admin.8)**:

```
# xfs_admin -O bigtime=1 device
```

Or with **xfs_repair(8) (https://man.archlinux.org/man/xfs_repair.8)**:

```
# xfs_repair -c bigtime=1 device
```

While there, you may want to enable `inobtcount` as well (another new default).

3 Performance

From **XFS FAQ (https://xfs.org/index.php/XFS_FAQ#Q:_I_want_to_tune_my_XFS_filesystems_for_.3Csomething.3E)**:

"The default values already used are optimised for best performance in the first place. mkfs.xfs will detect the difference between single disk and MD/DM RAID setups and change the default values it uses to configure the filesystem appropriately."

"In most cases, the only thing you need to to consider for `mkfs.xfs` is specifying the stripe unit and width for hardware RAID devices." (see **#Stripe size and width**)

> **Tip:** When using the XFS filesystem on **RAID** devices, performance improvements may be possible by using `largeio`, `swalloc`, increased `logbsize` and `allocsize` values, etc. The following articles may provide additional details about those flags:
>
> - **https://www.beegfs.io/wiki/StorageServerTuning**
> - **https://help.marklogic.com/Knowledgebase/Article/View/505/0/recommended-xfs-settings-for-marklogic-server**

"For mount options, the only thing that will change metadata performance considerably is the `logbsize` mount option. Increasing `logbsize` reduces the number of journal IOs for a given workload. The trade off for this increase in metadata performance is that more operations may be "missing" after recovery if the system crashes while actively making modifications."

> **Tip:** See **`xfs(5) (https://man.archlinux.org/man/xfs.5)`** for details on all available mount options.

"As of kernel 3.2.12, the default i/o scheduler, CFQ, will defeat much of the parallelization in XFS."

> **Note:** Arch is configured to use no I/O scheduler when a SATA or **NVMe** **SSD** is detected; this can confirmed by reading the content of `/sys/block/nvme*n*/queue/scheduler`.

Therefore for optimal performance, in most cases you can just follow **#Creation**.

3.1 Stripe size and width

If this filesystem will be on a striped RAID you can gain significant speed improvements by specifying the stripe size to the **`mkfs.xfs(8) (https://man.archlinux.org/man/mkfs.xfs.8)`** command.

XFS can sometimes detect the geometry under software RAID, but in case you reshape it or you are using hardware RAID see **how to calculate the correct sunit,swidth values for optimal performance (https://xfs.org/index.php/XFS_FAQ#Q:_How_to_calculate_the_correct_sunit.2Cswidth_values_for_optimal_performance)**.

3.2 Access time

On some filesystems you can increase performance by adding the `noatime` mount option to the `/etc/fstab` file. For XFS filesystems "the default atime behaviour is `relatime`, which has almost no overhead compared to noatime but still maintains sane atime values. All Linux filesystems use this as the default now (since around 2.6.30), but XFS has used relatime-like behaviour since 2006, so no-one should really need to ever use noatime on XFS for performance reasons."**[8] (https://xfs.org/index.php/XFS_FAQ#Q:_Is_using_noatime_or.2Fand_nodiratime_at_mount_time_giving_any_performance_benefits_in_xfs_.28or_not_using_them_performance_decrease.29.3F)**

See **Fstab#atime options** for more on this topic.

3.3 Discard

Despite XFS supporting async discard[9] (https://lwn.net/Articles/787272/) since kernel 4.7[10] (https://www.phoronix.com/scan.php?page=news_item&px=Async-Discard-Linux-4.7)[11] (https://events.static.linuxfound.org/sites/events/files/slides/discard_0.pdf), `xfs(5)` (https://man.archlinux.org/man/xfs.5) still recommends "that you use the **fstrim** application to discard unused blocks rather than the discard mount option because the performance impact of this option is quite severe."

See **Solid state drive#Periodic TRIM**.

3.4 Defragmentation

Although the extent-based nature of XFS and the delayed allocation strategy it uses significantly improves the file system's resistance to fragmentation problems, XFS provides a filesystem defragmentation utility (*xfs_fsr*, short for XFS filesystem reorganizer) that can defragment the files on a mounted and active XFS filesystem. It can be useful to view XFS fragmentation periodically.

`xfs_fsr(8)` (https://man.archlinux.org/man/xfs_fsr.8) improves the organization of mounted filesystems. The reorganization algorithm operates on one file at a time, compacting or otherwise improving the layout of the file extents (contiguous blocks of file data).

3.4.1 Inspect fragmentation levels

To see how much fragmentation your file system currently has:

```
# xfs_db -c frag -r /dev/partition
```

3.4.2 Perform defragmentation

To begin defragmentation, use the `xfs_fsr(8)` (https://man.archlinux.org/man/xfs_fsr.8) command:

```
# xfs_fsr /dev/partition
```

3.5 Deduplication

The *reflink* feature, available since kernel version 4.9 and enabled by default since `mkfs.xfs` version 5.1.0, allows creating fast reflink'ed copies of files as well as deduplication after the fact, in the same way as **btrfs**:

3.5.1 Reflink copies

Reflink copies initially use no additional space:

```
$ cp --reflink bigfile1 bigfile2
```

Until either file is edited, and a copy-on-write takes place. This can be very useful to create snapshots of (large) files.

3.5.2 Deduplication

Existing filesystems can be deduped using tools like **duperemove (https://archlinux.org/packages/?name=duperemove)**.

3.6 External XFS Journal

Using an external log (metadata journal) on for instance a **SSD** may be useful to improve performance **[12] (https://docs.oracle.com/en/operating-systems/oracle-linux/9/fsadmin/fsadmin-ManagingtheXFSFileSystem.html#extjnl-xfs)**. See **mkfs.xfs(8) (https://man.archlinux.org/man/mkfs.xfs.8)** for details about the `logdev` parameter.

> **Note:** Using flash-memory may wear-out the drive. See **Improving performance#Reduce disk reads/writes** for SSD wear-out details.

To reserve an external journal with a specified size when you create an XFS file system, specify the `-l logdev=`*`device`*`,size=`*`size`* option to the `mkfs.xfs` command. If you omit the `size` parameter, a journal size based on the size of the file system is used. To mount the XFS file system so that it uses the external journal, specify the `-o logdev=`*`device`* option to the **mount** command.

3.7 Sync interval

XFS has a dedicated **sysctl** variable for setting the **writeback interval** with a default value of 3000.

> **Warning:** While larger values may increase performance, they also increase the severity of data loss caused by a power outage.

```
/etc/sysctl.d/20-xfs-sync-interval.conf
```

```
fs.xfs.xfssyncd_centisecs = 10000
```

4 Administration

4.1 Resize

> **Note:** Currently, it is **not possible (https://xfs.org/index.php/Shrinking_Support)** to shrink XFS.

XFS can be resized online, after the partition has been altered. Just run `xfs_growfs` with the mount point as first parameter to grow the XFS filesystem to the maximal size possible.

```
# xfs_growfs /path/to/mnt/point
```

4.2 Online Metadata Checking (scrub)

Warning: This program is **experimental**, which means that its behavior and interface could change at any time. See **xfs_scrub(8) (https://man.archlinux.org/man/xfs_scrub.8)**.

`xfs_scrub` asks the kernel to scrub all metadata objects in the XFS filesystem. Metadata records are scanned for obviously bad values and then cross-referenced against other metadata. The goal is to establish a reasonable confidence about the consistency of the overall filesystem by examining the consistency of individual metadata records against the other metadata in the filesystem. Damaged metadata can be rebuilt from other metadata if there exists redundant data structures which are intact.

Enable/**start** `xfs_scrub_all.timer` to periodic check online metadata for all XFS filesystems.

Note: One may want to **edit** `xfs_scrub_all.timer` : the timer runs every Sunday at 3:10am and will be **triggered immediately** if it missed the last start time, i.e. due to the system being powered off.

4.3 Repair

Note: "Unlike other Linux file systems, *xfs_repair* does not run at boot time, even when an XFS file system was not cleanly unmounted. In the event of an unclean unmount, *xfs_repair* simply replays the log at mount time, ensuring a consistent file system."**[13] (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/xfsrepair)**

From **Checking and Repairing an XFS File System (https://docs.oracle.com/en/operating-systems/oracle-linux/6/adminsg/ol_repair_xfs.html)**:

"If you cannot mount an XFS file system, you can use the **xfs_repair -n** command to check its consistency. Usually, you would only run this command on the device file of an unmounted file system that you believe has a problem. The **xfs_repair -n** command displays output to indicate changes that would be made to the file system in the case where it would need to complete a repair operation, but will not modify the file system directly."

"If you can mount the file system and you do not have a suitable backup, you can use **xfsdump** to attempt to back up the existing file system data, However, the command might fail if the file system's metadata has become too corrupted."

"You can use the **xfs_repair** command to attempt to repair an XFS file system specified by its device file. The command replays the journal log to fix any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, it is usually not necessary to use the command, as the journal is replayed every time that you mount an XFS file system."

First **unmount** the filesystem, then run the **xfs_repair(8) (https://man.archlinux.org/man/xfs_repair.8)** tool:

```
# xfs_repair device
```

"If the journal log has become corrupted, you can reset the log by specifying the **-L** option **to xfs_repair**."

Warning: "The *xfs_repair* utility cannot repair an XFS file system with a dirty log. To clear the log, mount and unmount the XFS file system. If the log is corrupt and cannot be replayed, use the `-L` option ("force log zeroing") to clear the log, that is, `xfs_repair -L /dev/device`. Be aware that this may result in further corruption or data loss."**[14] (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/storage_administration_guide/xfsrepair)**

Warning: "Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you are experienced in debugging and repairing XFS file systems using **xfs_db**, it is recommended that you instead recreate the file system and restore its contents from a backup."**[15] (https://docs.oracle.com/en/operating-systems/oracle-linux/6/adminsg/ol_repair_xfs.html)**

"If you cannot mount the file system or you do not have a suitable backup, running **xfs_repair** is the only viable option unless you are experienced in using **xfs_db**."

"**xfs_db** provides an internal command set that allows you to debug and repair an XFS file system manually. The commands allow you to perform scans on the file system, and to navigate and display its data structures. If you specify the **-x** option to enable expert mode, you can modify the data structures."

```
# xfs_db [-x] device
```

"For more information, see the **`xfs_db(8)` (https://man.archlinux.org/man/xfs_db.8)** and **`xfs_repair(8)` (https://man.archlinux.org/man/xfs_repair.8)**, and the **help** command within **xfs_db**."

See also **Which factors influence the memory usage of xfs_repair? (https://xfs.org/index.php/XFS_FAQ#Q:_Which_factors_influence_the_memory_usage_of_xfs_repair.3F)** and **XFS Repair (https://xfs.org/docs/xfsdocs-xml-dev/XFS_User_Guide/tmp/en-US/html/xfs-repair.html)**.

4.4 Data rescue

Even when being mounted read-only with `mount -o ro` an XFS file system's log will be replayed if it has not been unmounted cleanly.

There may be situations where a compromised XFS file system on a damaged storage device should be mounted read-only, so that files may be copied off it hopefully without causing further damage, yet it cannot be mounted because it has not been unmounted cleanly and is damaged to such an extent that the log cannot be replayed. Also, consider that replaying the log means writing to the compromised file system, which might be a bad idea in itself.

To mount an XFS file system without writing to it in any way and without replaying the log, use `mount -o ro,norecovery`.

4.5 Undelete

`xfs_undelete` (https://aur.archlinux.org/packages/xfs_undelete/)[AUR] can recover (under certain conditions) deleted files on an unmounted or read-only mounted XFS filesystem. See https://github.com/ianka/xfs_undelete for more information.

5 Troubleshooting

5.1 Root file system quota

XFS quota mount options (`uquota`, `gquota`, `prjquota`, etc.) fail during re-mount of the file system. To enable quota for root file system, the mount option must be passed to initramfs as a kernel parameter `rootflags=`. Subsequently, it should not be listed among mount options in `/etc/fstab` for the root (`/`) filesystem.

> **Note:** There are some differences of XFS Quota compared to standard Linux Disk quota, this article https://inai.de/linux/adm_quota may be worth reading.

5.2 xfs_scrub_all fails if user "nobody" can not access the mountpoint

When running `xfs_scrub_all`, it will launch `xfs_scrub@.service` for each mounted XFS file system. The service is run as user `nobody`, so if `nobody` can not navigate to the directory, it will fail with the error:

```
xfs_scrub@mountpoint.service: Changing to the requested working directory failed: Permission denied
xfs_scrub@mountpoint.service: Failed at step CHDIR spawning /usr/bin/xfs_scrub: Permission denied
xfs_scrub@mountpoint.service: Main process exited, code=exited, status=200/CHDIR
```

To allow the service to run, change the permissions of the mountpoint so that user `nobody` has execute permissions.

5.3 fsck.xfs fails in systemd-based initramfs

When using a mkinitcpio-generated systemd based initramfs without the `base` hook, you will see the following messages in the journal:

```
systemd-fsck[288]: fsck: /usr/bin/fsck.xfs: execute failed: No such file or directory
systemd-fsck[286]: fsck failed with exit status 8.
systemd-fsck[286]: Ignoring error.
```

This is because `fsck.xfs(8)` (https://man.archlinux.org/man/fsck.xfs.8) is a shell script and requires `/bin/sh` to execute. `/usr/bin/sh` is provided by the `base` hook, so the solution is to prepend it to the HOOKS array in `/etc/mkinitcpio.conf`. E.g.:

```
HOOKS=(base systemd ... )
```

6 See also

- XFS wiki (https://xfs.wiki.kernel.org/)
- XFS FAQ (https://xfs.org/index.php/XFS_FAQ)
- Improving Metadata Performance By Reducing Journal Overhead (https://xfs.org/index.php/Improving_Metadata_Performance_By_Reducing_Journal_Overhead)
- XFS Wikipedia Entry
- XFS User Guide (https://xfs.org/index.php/XFS_Papers_and_Documentation) XFS User Guide no longer exists but has a link to the git repository

Retrieved from "https://wiki.archlinux.org/index.php?title=XFS&oldid=756945"