Sparse Files

Christian Külker

2022-05-20

Contents

Sparse files are files that allocate data to disk space and summarize empty space (null bytes blocks) in the meta data, making the use of the file system storage more efficiently. When meta data space is allocated to data, the data is written to the file system and the count of empty space is reduced. In this manner reported file size and actual used disk space differs for sparse files that contain empty space. The size of the file gets its maximum on the file system when there is no empty data. Therefore sparse files makes only sense for information that has a lot of empty data or at least a lot of empty data in the beginning. It makes no sense to create sparse file to be filled completely with data.

The ability to creates sparse files is dependent on the file system. A peculiar feature of sparse files is the discrepancy with reported size (**apparent size**) and **actual size** of used blocks on the storage media, which makes the use of some command not intuitive. This document shows how to create sparse file with different tools as well as measure them.

Reading sparse file will convert transparently to the application empty meta data to empty blocks filled with zero bytes. This is the opposite when writing. On Linux most modern file systems support sparse files. That includes NTFS but not `HFS+`.

A very common usage for sparse files are disk images. A disk image is a large file containing data. Often the data is a formatted file system. Usually new file systems are mostly empty, therefore using sparse files for this purpose safes a large amount of real disk space until the disk image file system is filled with non empty data. However it has to be noted that sparse files are not a miracle. If a sparse file is created that is larger than the actual free space of a disk, which is possible, it will grow up to the remaining empty disk space, but not beyond. Of course planning ahead is important as running a file system on disk or inside a sparse file that hits the boundaries of free space is a nightmare as the kind of corruption that may occur is not predictable.

While the creation of a sparse file is very efficient, when used as disk image a sparse file can become fragmented and less efficient. Filling up file systems inside a sparse file can have unexpected effects. Of course some tools will not show that a certain file is a sparse file. And when copied or moved around the tool to do the movement or copy need to be capable of handling sparse files. If a too is used that can not copy a sparse file, the empty data represented in the meta data of the file system can be omitted and the result will be a corrupted file or a file which content is not usable any more. In a minor severe case a sparse file would be copied to a non sparse file with lot of empty space allocated to the disk, which would revert the usefulness of using sparse files. That might happen when using sparse file aware tool to copy a sparse file from a sparse file aware file system to a non sparse file aware file system.

The following file systems for Linux (according to Wikipedia File Systems) might support sparse files:

- BeeGFS

- APFS
- V6FS, V7FS
- GFS
- NTFS
- FFS
- UFS1, UFS 2
- LFS
- ext, ext2, ext3, ext4
- NOVA
- F2FS
- Lustre
- NILFS
- ReiserFS, Reiser4
- OCFS2
- XFS
- JFS
- VxFS
- UDF
- ZFS
- Btrfs
- ReFS
- SquashFS
- BlueStore/Cephfs

1 Making

Each of the below two commands create a 128MB sparse file called `file.image`.

```
truncate -s 128M file.image
dd if=/dev/zero of=file.image bs=1 count=0 seek=128M
```

2 Measure

2.1 Apparent Size

```
du -h --apparent file.image
128M    file.image
```

2.2 Actual Size

```
du -h file.image
0    file.image
```

2.3 Apparent And Actual Size

```
du -h file.image;du -h --apparent file.image
0       file.image
128M    file.image
  # Shows first current size (0) and apparent size (128M)
ls -hls file.image
0 -rw-r--r-- 1 root root 128M Mar 21 14:09 file.image
  # The same for a used sparse file
ls -hls file.image
4.1M -rw-r--r-- 1 root root 1.0G Mar 21 13:20 file.image
```

Some `du` use `--apparent-size` instead of `--apparent`.

3 Converting

3.1 To

Converting files to sparse files works only on supported file systems.

```
  # make a non sparse file
cp unknown.image non-sparse.image --sparse=never
du -h non-sparse.image
128M    non-sparse.image
  # convert a non sparse file to a sparse file
fallocate -d non-sparse.image
  # This step is not need, but would remind about the feature
mv non-sparse.image sparse.image
du -h sparse.image
0    sparse.image
```

3.2 From

```
cp sparse.image non-sparse.image --sparse=never
```

4 Copying

Usually `cp` can detect a sparse file. No option is needed. However if one want to be explicit the `--sparse` option can be used. This can also be used convert a non sparse file to a sparse file while copying. When using `rsync` the `-S` or `--sparse` option need to be set.

```
cp old.image new.image # sparseness of files is the same
cp --sparse=always old.image new.image # old.image could be non-sparse
rsync -S old.image new.image
```

5 Archiving

Per default `tar` uses non-sparse files. And `tar` converts sparse file to non-sparse files! To use `tar` and keep the sparse feature of files use the `-S` option.

This example used the sparse file `file.image` that contains a `XFS` file system that uses 3.6M of meta data (that is of course not empty and therefore not sparse).

```
tar -cf file.tar file.image
du -h file.tar --apparent; du -h file.tar
129M    file.tar
129M    file.tar

tar -Scf file.tar file.image
du -h file.tar --apparent; du -h file.tar
3.6M    file.tar
3.6M    file.tar
```

6 Creating A File System Inside

```
mkfs.xfs sparse.image
```

```
1  meta-data=file.image             isize=512    agcount=4, agsize=8192 blks
2           =                       sectsz=512   attr=2, projid32bit=1
3           =                       crc=1        finobt=1, sparse=0, rmapbt=0,
       reflink=0
4  data     =                       bsize=4096   blocks=32768, imaxpct=25
5           =                       sunit=0      swidth=0 blks
6  naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
7  log      =internal log           bsize=4096   blocks=855, version=2
8           =                       sectsz=512   sunit=0 blks, lazy-count=1
```

```
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

```
du -h file.image --apparent; du -h file.image
```

```
128M    file.image
3.6M    file.image
```

7 Enlarge

With `dd` it is possible to grow an existing sparse file. Even if it contains a file system.

```
dd if=/dev/zero of=file.image bs=1 count=0 seek=1G
du -h file.image --apparent; du -h file.image
1.0G    file.image
3.6M    file.image
```

8 Mounting A File System

There is no difference of mounting file systems of non-sparse files and sparse files.

```
mkdir mountpoint
mound -o loop file.image mountpoint
df -h|grep mountpoint
/dev/loop0               125M  6.8M  118M   6% /tmp/mountpoint
```

9 Resize A File System

Resizing a file system is a delicate matter. As a matter of fact the way to resize a file system depends on the file system and its tools. Just to see how it is done in this section should not let you assume that it is done similar to a different file system. Also be warned, resizing a used file system **is risky** and data loss may occur. Better make a validated backup before trying to resize a partition.

While a sparse file containing `riserfs` can (and should?) be resized while it is **not** mounted. A `XFS` can only be resized while it **is** mounted.

Action	XFS	ReiserFS
create sparse file	not mounted	not mounted
make file system	not mounted	not mounted

Action	XFS	ReiserFS
grow sparse file	not mounted	not mounted
grow file system	mounted	not mounted

The command to grow a file system depends on the file system. For `xfs` it is `xfs_growfs` for `reiserfs` it is `resize_reiserfs`.

```
  # Create a 128 MB image
truncate -s 128M file.image
du -h file.image;du --apparent -h file.image
0       file.image
128M    file.image
```

```
  # Make a XFS file system
mkfs.xfs file.image
meta-data=file.image             isize=512    agcount=4, agsize=8192 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=0, rmapbt=0,
 ↪  reflink=0
data     =                       bsize=4096   blocks=32768, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal log           bsize=4096   blocks=855, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
du -h file.image;du --apparent -h file.image
3.6M    file.image
128M    file.image
```

```
  # Enlarge the XFS file system from 128MB to 1GB
dd if=/dev/zero of=file.image bs=1 count=0 seek=1G
0+0 records in
0+0 records out
0 bytes copied, 4.7421e-05 s, 0.0 kB/s
du -h file.image;du --apparent -h file.image
3.6M    file.image
1.0G    file.image
```

```
  # Mount XFS file system (a mount point will be created in /tmp/SPARSE)
mkdir mnt
mount -o loop file.image mnt
```

```
df -h|grep mnt
/dev/loop0               125M  6.8M  118M   6% /tmp/SPARSE/mnt
```

```
  # Grow the XFS file system
xfs_growfs mnt
meta-data=/dev/loop0             isize=512    agcount=4, agsize=8192 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1 spinodes=0 rmapbt=0
         =                       reflink=0
data     =                       bsize=4096   blocks=32768, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=1
log      =internal               bsize=4096   blocks=855, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 32768 to 262144
du -h file.image;du --apparent -h file.image
4.1M    file.image
1.0G    file.image
df -h|grep mnt
/dev/loop0              1021M  9.2M 1012M   1% /tmp/SPARSE/mnt
```

```
  # Unmount the file system
umount mnt
du -h file.image;du --apparent -h file.image
4.1M    file.image
1.0G    file.image
```

10 Links

- Wikipedia Sparse File
- Wikipedia File Systems

11 History

Version	Date	Notes
0.1.1	2022-05-20	Improve shell blocks, typos
0.1.0	2022-03-21	Initial release

12 Disclaimer of Warranty

THERE IS NO WARRANTY FOR THIS INFORMATION, DOCUMENTS AND PROGRAMS, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE INFORMATION, DOCUMENT OR THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE INFORMATION, DOCUMENTS AND PROGRAMS IS WITH YOU. SHOULD THE INFORMATION, DOCUMENTS OR PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

13 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.