Sparse Files

Christian Külker

2023-03-10

Contents

1 Making . . . 3

2 Measure . . . 3
2.1 Apparent Size . . . 3
2.2 Actual Size . . . 4
2.3 Apparent And Actual Size . . . 4

3 Converting . . . 4
3.1 To . . . 4
3.2 From . . . 4

4 Copying . . . 5

5 Archiving . . . 5

6 Creating A File System Inside . . . 5

7 Enlarge . . . 6

8 Mounting A File System . . . 6

9 Resize A File System . . . 6

10 Links . . . 8

11 History . . . 8

12 Disclaimer of Warranty . . . 9

13 Limitation of Liability . . . 9

Sparse files__ are files that allocate data to disk space and aggregate empty space (blocks of zero bytes) in the metadata to make more efficient use of file system space. When metadata space is allocated to data, the data is written to the file system and the amount of empty space is reduced. In this way, the reported file size and the actual disk space used differ for sparse files that contain empty space. The size of the file on the file system reaches its maximum when there is no empty data. Therefore, sparse files only make 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 a sparse file to be completely filled with data.

The ability to create sparse files depends on the file system. A peculiarity of sparse files is the discrepancy between the reported size (**apparent size**) and the **actual size** of used blocks on the storage media, which makes the use of some commands not intuitive. This document shows how to create sparse files using various tools and how to measure them.

When reading a sparse file, empty metadata is transparently converted to empty blocks filled with zero bytes. Writing does the opposite. On Linux, most modern file systems support sparse files. This includes NTFS, but not `HFS+`.

A very common use for sparse files is disk images. A disk image is a large file containing data. The data is often a formatted file system. Usually new file systems are mostly empty, so using sparse files for this purpose saves a lot of real disk space until the disk image file system is filled with non-empty data. It should be noted, however, that sparse files are no 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 free space, but not beyond. Of course, planning ahead is important, as running a file system on a disk or within a sparse file that hits the limits of free space is a nightmare, as the type of corruption that can occur is unpredictable.

While creating a sparse file is very efficient, when used as a disk image, a sparse file can become fragmented and less efficient. Filling file systems inside a sparse file can have unexpected effects. Of course, some tools will not indicate that a given file is a sparse file. And when copying or moving, the tool to do the moving or copying needs to be able to handle sparse files. If a tool is used that cannot copy a sparse file, the empty data represented in the metadata of the file system may be omitted, and the result will be a corrupted file or a file whose contents are no longer usable. In a less severe case, a sparse file would be copied to a nonsparse file with a lot of empty space allocated to the disk, which would negate the usefulness of using sparse files. This can happen when using a 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) may 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 two commands below creates 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 only works 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 recognize a sparse file. No option is needed. However, if you want to be explicit, the `--sparse` option can be used. This can also be used to convert a non-sparse file into a sparse file during copying. When using `rsync` the `-S` or `--sparse` option must 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

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

This example used the sparse file `file.image` which contains an `XFS` file system that uses 3.6M of metadata (which 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
```

```
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 --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 filesystem.

```
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 between mounting non-sparse and sparse file systems.

```
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 tricky business. In fact, the way to resize a filesystem depends on the filesystem and its tools. Just to see how it is done in this section should not make you assume it is done similarly on another file system. Also be warned, resizing a used filesystem is **risky** and data loss can 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. An `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
grow sparse file	not mounted	not mounted
grow file system	mounted	not mounted

The command to grow a filesystem depends on the filesystem. 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
0.1.2	2023-03-10	Improve writing
0.1.1	2022-05-20	Improve shell blocks, typos
0.1.0	2022-03-21	Initial release

12 Disclaimer of Warranty

13 Limitation of Liability