XFS Practical Exercises

02 – mkfs & mount

Overview

In order to use XFS we must first create a filesystem. Filesystems can be used for various different purposes and the tool used to create an XFS filesystem, mkfs.xfs, can be used to customise the characteristics of the filesystem.

Goals

The goal of this lab is to understand the procedure for creating an XFS filesystem including the options available for customising it for a particular purpose.

In this lab exercise we will explore the options to:

- adjust the filesystem block size
- filesystem directory block size
- set the stripe unit and width sizes
- configure an external log
- set the allocation group count and/or size

Prerequisites

It is assumed that the reader is familiar with basic UNIX commands, in particular mkfs and mount.

Setup

- Recent Linux kernel with XFS enabled (and loaded if installed as a module)
- XFS user-space commands package, xfsprogs, installed.
- One or more spare block device(s).

Exercises

Exercise 1

1. Make an XFS filesystem using default options. Note: the -f option is required to force mkfs to overwrite an existing filesystem.

```
# mkfs -t xfs -f /dev/sda4

meta-data=/dev/sda4              isize=256    agcount=16, agsize=461617 blks

         =                       sectsz=512   attr=0

data     =                       bsize=4096   blocks=7385872, imaxpct=25

         =                       sunit=0      swidth=0 blks, unwritten=1

naming   =version 2              bsize=4096

log      =internal log           bsize=4096   blocks=3606, version=1

         =                       sectsz=512   sunit=0 blks

realtime =none                   extsz=65536  blocks=0, rtextents=0
```

2. Mount the filesystem:

```
# mkdir mnt

# mount /dev/sda4 /mnt

# mount

/dev/sda2 on / type xfs (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

tmpfs on /dev/shm type tmpfs (rw)

devpts on /dev/pts type devpts (rw,mode=0620,gid=5)

usbfs on /proc/bus/usb type usbfs (rw)

/dev/sda4 on /mnt type xfs (rw)
```

3. Query the filesystem's parameters:

```
# xfs_info /mnt

meta-data=/dev/sda4              isize=256    agcount=16, agsize=461617 blks

         =                       sectsz=512   attr=0
```

```
data      =                       bsize=4096   blocks=7385872, imaxpct=25
          =                       sunit=0      swidth=0 blks, unwritten=1
naming    =version 2              bsize=4096
log       =internal               bsize=4096   blocks=3606, version=1
          =                       sectsz=512   sunit=0 blks
realtime  =none                   extsz=65536  blocks=0, rtextents=0
```

4. Unmount the filesystem:

```
# umount /mnt
```

Exercise 2

Make an XFS filesystem to contain many small files per directory

- reduce filesystem block size to 512 bytes to save wasted space
- use the maximum directory block size of 4KB (this is actually the default), this allows for more entries per level in the directory b-trees resultling in faster lookups.

1. Create the filesystem.

```
# mkfs -t xfs -f -b size=512 -n size=4096 /dev/sda4
meta-data=/dev/sda4               isize=256    agcount=16, agsize=3692941 blks
          =                       sectsz=512   attr=0
data      =                       bsize=512    blocks=59087056, imaxpct=25
          =                       sunit=0      swidth=0 blks, unwritten=1
naming    =version 2              bsize=4096
log       =internal log           bsize=512    blocks=28851, version=1
          =                       sectsz=512   sunit=0 blks
realtime  =none                   extsz=65536  blocks=0, rtextents=0
# mount /dev/sda4 /mnt
# cd /mnt
# for file in `seq 0 1000`; do echo $file > $file; done
# du -k
537     .
```

2. Try the same test with a default block size of 4KB and verify the savings for yourself.

Exercise 3

Make an XFS filesystem that is aligned on stripe width/unit boundaries:

- use a stripe unit size of 128KB (Note: when using the sunit option the value is in 512 byte blocks)
- use a stripe width size of 2MB (Note: when using the swidth option the value is in 512 byte blocks and is a multiple of the sunit value)

1. Create the stripe aligned filesystem.

```
# mkfs -t xfs -f -d sunit=256,swidth=2048 /dev/sda4

meta-data=/dev/sda4              isize=256    agcount=16, agsize=461632 blks

         =                       sectsz=512   attr=0

data     =                       bsize=4096   blocks=7385872, imaxpct=25

         =                       sunit=32     swidth=256 blks, unwritten=1

naming   =version 2              bsize=4096

log      =internal log           bsize=4096   blocks=3616, version=1

         =                       sectsz=512   sunit=0 blks

realtime =none                   extsz=65536  blocks=0, rtextents=0
```

The sunit and swidth values displayed by mkfs are in filesystem blocks (ie 32 * 4KB = 128KB)

Exercise 4

Make an XFS filesystem that uses an external log on a second device. Note: the maximum size for a log is 128MB, if the log device is larger than this the size option must be specified.

1. Create the filesystem with an external log.

```
# mkfs -t xfs -f -l logdev=/dev/sda3,size=128m /dev/sda4

meta-data=/dev/sda4              isize=256    agcount=16, agsize=461617 blks

         =                       sectsz=512   attr=0

data     =                       bsize=4096   blocks=7385872, imaxpct=25

         =                       sunit=0      swidth=0 blks, unwritten=1

naming   =version 2              bsize=4096

log      =/dev/sda3              bsize=4096   blocks=32768, version=1

         =                       sectsz=512   sunit=0 blks

realtime =none                   extsz=65536  blocks=0, rtextents=0
```

2. In order to mount a filesystem with an external log, the log device must be specified on the mount command line.

```
# mount -o logdev=/dev/sda3 /dev/sda4 /mnt

# mount

/dev/sda2 on / type xfs (rw)

proc on /proc type proc (rw)
```

```
sysfs on /sys type sysfs (rw)

tmpfs on /dev/shm type tmpfs (rw)

devpts on /dev/pts type devpts (rw,mode=0620,gid=5)

usbfs on /proc/bus/usb type usbfs (rw)

/dev/sda4 on /mnt type xfs (rw,logdev=/dev/sda3)
```

3. If the log device is not specified the filesystem will fail to mount:

```
# mount /dev/sda4 /mnt

mount: wrong fs type, bad option, bad superblock on /dev/sda4,

       missing codepage or other error

       In some cases useful info is found in syslog - try

       dmesg | tail
```

Exercise 5

Make an XFS filesystem with more allocation groups to allow more parallelism when allocating blocks and inodes. Increasing the number of allocation groups will decrease the space available in each group. For most workloads, filesystem configurations with a very small or very large number of allocation groups should be avoided. The capacity of your block device may cause mkfs.xfs to report different values for agcount and agsize. For this exercise take the default agcount value and double it.

1. Create a simple filesystem first.

```
# mkfs -t xfs -f /dev/sda4

meta-data=/dev/sda4              isize=256    agcount=16, agsize=461617 blks

         =                       sectsz=512   attr=0

data     =                       bsize=4096   blocks=7385872, imaxpct=25

         =                       sunit=0      swidth=0 blks, unwritten=1

naming   =version 2              bsize=4096

log      =internal log           bsize=4096   blocks=3606, version=1

         =                       sectsz=512   sunit=0 blks

realtime =none                   extsz=65536  blocks=0, rtextents=0
```

2. In this example. the default agcount is 16 so let's try 32.

```
# mkfs -t xfs -f -d agcount=32 /dev/sda4

meta-data=/dev/sda4              isize=256    agcount=32, agsize=230809 blks

         =                       sectsz=512   attr=0

data     =                       bsize=4096   blocks=7385883, imaxpct=25

         =                       sunit=0      swidth=0 blks, unwritten=1

naming   =version 2              bsize=4096
```

```
log      =internal log          bsize=4096   blocks=3606, version=1

         =                      sectsz=512   sunit=0 blks

realtime =none                  extsz=65536  blocks=0, rtextents=0
```

3. Or alternatively, set the allocation group size. In this example set each allocation group to 4GB. If the capacity of your block device is less than 4GB mkfs will fail, in this case pick a value for agsize that is one quarter of the device's capacity.

```
# mkfs -t xfs -f -d agsize=4g /dev/sda4

meta-data=/dev/sda4             isize=256    agcount=8, agsize=1048576 blks

         =                      sectsz=512   attr=0

data     =                      bsize=4096   blocks=7385883, imaxpct=25

         =                      sunit=0      swidth=0 blks, unwritten=1

naming   =version 2             bsize=4096

log      =internal log          bsize=4096   blocks=3606, version=1

         =                      sectsz=512   sunit=0 blks

realtime =none                  extsz=65536  blocks=0, rtextents=0
```

Questions

1. From the following information calculate the stripe unit and width:

```
meta-data=/dev/sda4              isize=256    agcount=16, agsize=923264 blks
         =                       sectsz=512   attr=0
data     =                       bsize=2048   blocks=14771760, imaxpct=25
         =                       sunit=128    swidth=256 blks, unwritten=1
naming   =version 2              bsize=4096
log      =internal log           bsize=2048   blocks=7296, version=1
         =                       sectsz=512   sunit=0 blks
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

2. What would be the conseqence of not aligning a filesystem on a stripe unit correctly?
3. What are the advantages of an external log?
4. Why would having very few or very many allocation groups be a bad idea?

Answers

1. Stripe unit = 128 * 2048 = 256KB, stripe width = 256 * 2048 = 512KB
2. If a block write is not aligned on a RAID stripe unit boundary and is not a full stripe unit, the RAID will be forced to do a read/modify/write cycle to write the data. This can have a significant performance impact. By setting the stripe unit size properly, XFS will avoid unaligned accesses.
3. With an internal log, excessive amounts of log activity can adversely affect the performance of the rest of the filesystem. The external log could also be placed on higher speed storage.
4. Too few allocation groups limits the ability to parallelise concurrent disk block and/or inode allocations. Too many allocation groups (with little space in each one) will increase the probability of exhausting an allocation group and having to search for another group that has free space. This can cause an unreasonable amount of CPU time to be used when the filesystem is close to full.