Silicon Graphics, Inc.

XFS Overview & Internals
04 - mkfs

November 2006

Creating XFS Filesystems

- mkfs.xfs supports a large number of options for configuration a large number of different XFS filesystems
  - See mkfs.xfs(8)

- Units
  - `100s = 100 sectors = 100 x 512 bytes*`
  - `100b = 100 blocks = 100 x 4 kilobytes*`
  - `100k = 100 * 1024 bytes`
  - `Assuming 512 bytes sectors and 4 KB filesystem block`

- -N option can be used to show filesystem parameters without creating a filesystem
- Also provides the capability to pre-initialise the filesystem with directories and inodes, which is useful for testing

mkfs – Allocation Block Size

- Specify the fundamental block size of the filesystem.
- The default value is 4096 bytes (4 KB), the minimum is 512, and the maximum is 65536 (64 KB).
- XFS on Linux currently only supports pagesize or smaller blocks.
- To create a filesystem with a block size of 2048 bytes you would use:

```
mkfs -b size=2048 device
```

- *Smaller block sizes reduce wasted space for lots of small files.*

mkfs – Allocation groups

- The data section of an XFS filesystem is divided into allocation groups.
- More allocation groups imply more parallelism when allocation blocks and inodes.
- To create filesystem with 16 allocation groups you would use:

```
mkfs -d acount=16 device
```

- *To create a filesystem with fixed size allocation groups*

```
mkfs -d agsize=4g device
```

- *Filesystems with too few or too many allocation groups should be avoided.*

mkfs – Stripe Alignment

- Aligning file data on stripe width boundaries can significantly improve performance on large RAIDs
  - A 2MB write to filesystem with a 2MB stripe width and 512KB stripe unit will result in four I/Os, one to each lun. Without alignment this would often require two I/Os to one disk, and one I/O to the other three, taking longer than it should.

- To create a filesystem with a stripe unit of 1MB for an 8+1 RAID you would use:

```
mkfs -d sunit=2048,swidth=16384 device
```

or

```
mkfs -d su=1m,sw=8m device
```

mkfs – Unwritten Extents

- Unwritten extents are used to support pre-allocation.
- Default is enabled.
- To disable unwritten extents you would use:

```
mkfs -d unwritten=0 device
```

- *Filesystem write performance may be negatively affected for unwritten file extents, since extra filesystem transactions are required to convert extent flags for the range of the file written.*

mkfs – Inode Options

- The mkfs inode options allow the user to change
  - The size of the inode from the default 256 bytes to up 2048 bytes
  - The maximum number of inodes in the filesystem, defaults to 25%
  - Extended attribute allocation policy

- These options may impact
  - where the inode is allocated in filesystem
  - how much data needs to be read from disk to read the inode
  - how much data may be associated with the inode before data goes out of line

- To allocate 512 bytes to each inode:

```
mkfs -i size=512 device
```

mkfs – Extended Attributes

- Specify the version of extended attribute inline allocation policy to be used.
- Default is zero, when extended attributes are used for the first time the version will be set to either one or two.
- Version two uses a more efficient algorithm for managing the available inline inode space than version one, however, for backward compatibility, version one is selected by default.
- To force the use of version two extended attributes you would use:

```
mkfs -i attr=2 device
```

mkfs – Naming options

- Specify the version and size parameters for the naming (directory) area of the filesystem.
- The naming (directory) version is 1 or 2, defaulting to 2 if unspecified.
- XFS on Linux does not support naming (directory) version 1.
- To create a filesystem with a directory block size of 16KB you would use:

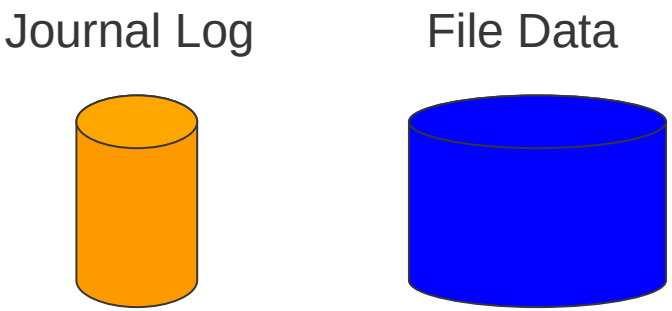
```
mkfs -n size=16384 device
```

mkfs – External Log

- The journal log can be on a different device to the rest of the filesystem
  - At least 512 blocks.
  - No more than 64K blocks or 128MB, whichever is smaller
  - Defaults to maximum size for >1TB filesystems
- Log device could be a device with better IOPS performance
  - 15K RPM disk or battery-backed memory

`mkfs.xfs -l logdev=`*`log_device device`*

`mount -o logdev=`*`log_device device path`*

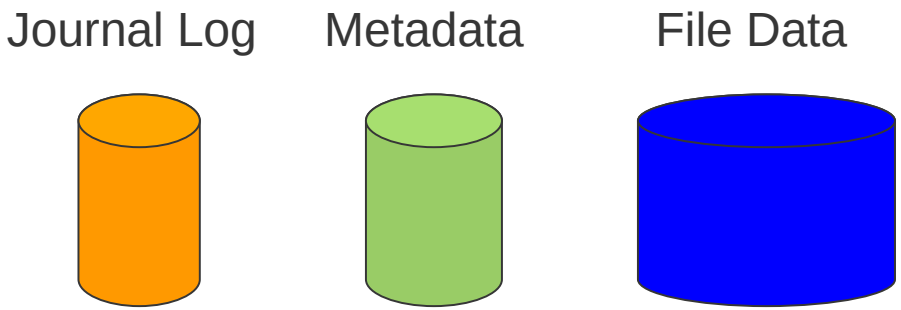


mkfs - Realtime

- Originally designed for media streaming requiring predictable latencies
  - Realtime volume tuned for large files without small inode clusters
- Metadata and file data on separate volumes
  - Bitmap allocator only has to manage file data allocations

```
mkfs.xfs -l logdev=log_device -r rtdev=rt_device device
mount -o logdev=log_device,rtdev=rt_device device path
```

- *rt_device* is the device for the file data, *device* is for the metadata
- Receives limited testing and support in Linux



mkfs – Filesystem Image

- mkfs allows you to create a filesystem as a regular file
- This can be used to create a filesystem on a loop-back device

```
mkfs -d file=1,name=filename,size=2g filename
```

mkfs – Sector Size

- Specifies the fundamental sector size of the filesystem.
- Default value is 512 bytes.
- The minimum value for sector size is 512; the maximum is 32768 (32 KB) or the filesystem block size (whichever is smaller).
- To create a filesystem on a device that has a sector size of 1KB you would use:

```
mkfs -s size=1024 device
```

mkfs – Filesystem Label

- Set the filesystem label.
- XFS filesystem labels can be at most 12 characters long.
- To create a filesystem and specify the label you would use:

```
mkfs -L label device
```

mkfs – Prototype Filesystem

- The –p option to mkfs allows you to specify a prototype file
- The prototype file lists inodes and their metadata that mkfs will add to the filesystem when it is created
- Since the filesystem is not mounted at this time no logging is required, so mkfs can create millions of inodes at a much faster rate than on a live filesystem
- This is mainly used for testing XFS when lots of inodes are required