



Silicon Graphics, Inc.

XFS Overview & Internals

02 - Overview

© Copyright 2006 Silicon Graphics Inc. All rights reserved.

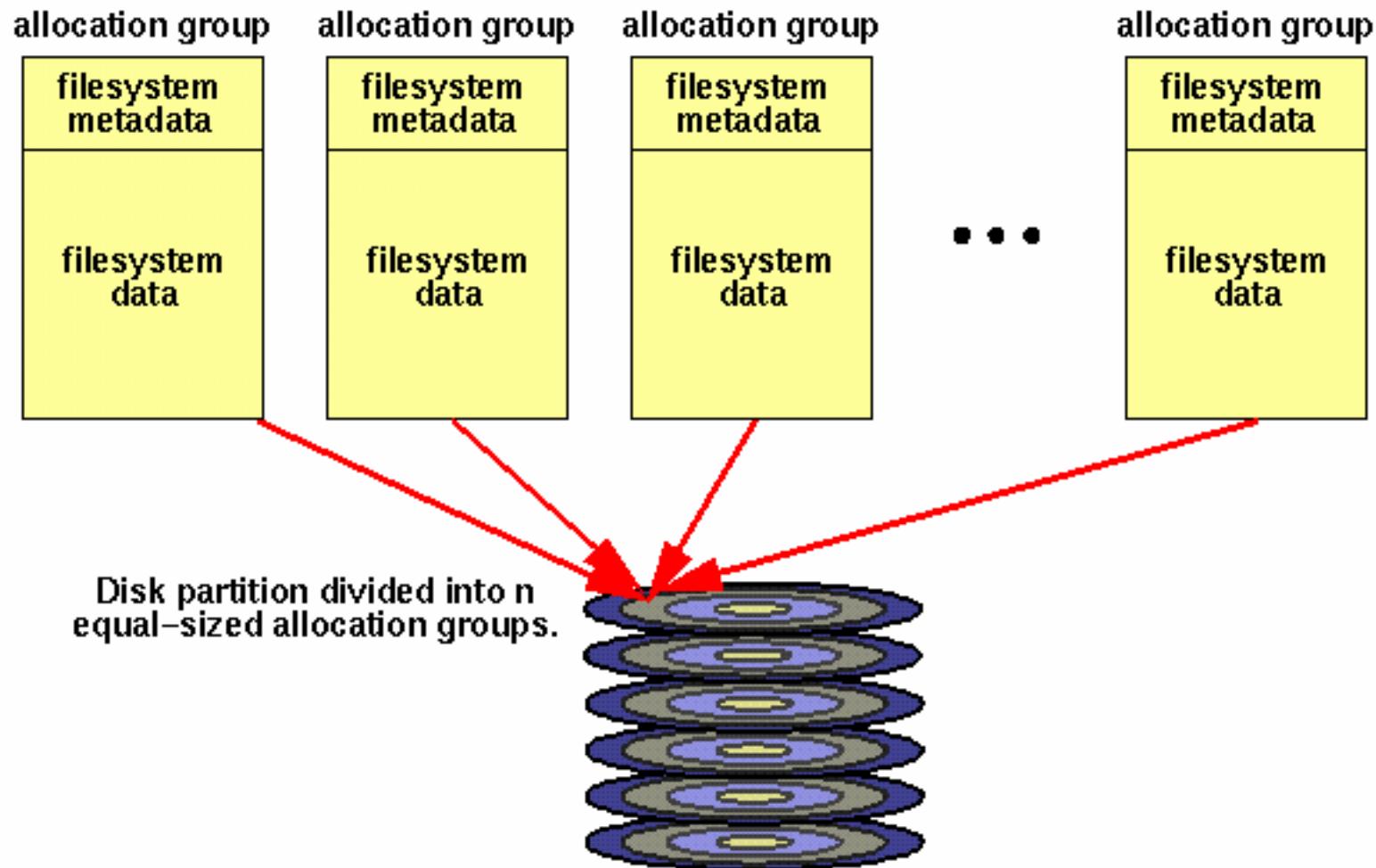
Permission is granted to copy, distribute, and/or modify this document under the terms of the Creative Commons Attribution-Share Alike, Version 3.0 or any later version published by the Creative Commons Corp. A copy of the license is available at <http://creativecommons.org/licenses/by-sa/3.0/us/> .

November 2006

XFS Filesystem Structure

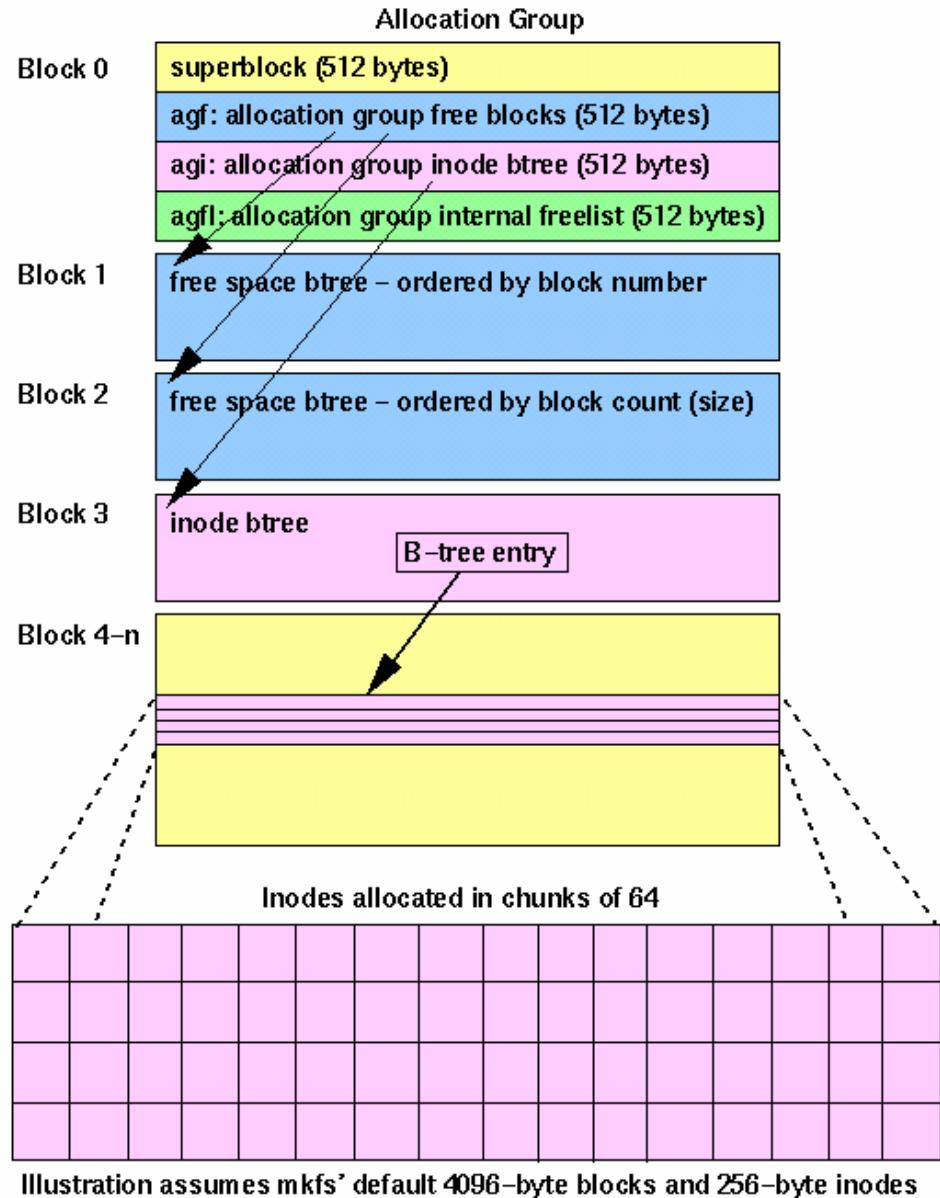
- This section gives an overview of the structure of an XFS filesystem
- More detailed examination of the filesystem structure is covered later in the course
- An XFS filesystem is divided evenly into allocation groups
- An allocation group can be from 16MB to 1TB in size
- See xfs(5)

Allocation Groups



Allocation Group Structure

- Each allocation group includes
 - Super block information about the entire filesystem
 - Free space management (within the allocation group)
 - Inode allocation and tracking (with the allocation group)
- Inode clusters within an allocation group are created when needed
 - mkfs.xfs does not pre-create inodes throughout the filesystem



XFS Limits

- 32 bit Linux
 - Maximum File Size = 16TB (O_LARGEFILE)
 - Maximum Filesystem Size = 16TB
- 64 bit Linux
 - Maximum File Size = 9 Million TB = 9 ExaB
 - Maximum Filesystem Size = 18 Million TB = 18 ExaB

Filesystem Block Size (FSB)

- Filesystem blocks (FSBs) are the unit of space for a filesystem
 - Filesystem blocks are comprised of one or more device-level sectors.
- The page management implementation in Linux limits the FSB size to the page size
 - 4KB on ia32 and x86_64 architectures
 - 16KB on ia64
- Performance can improve with different block sizes depending on the size of I/O requests and the size of files
 - Larger blocks will also use more disk space for small (<1FSB) files

Extents

- An extent is a set of one or more contiguous FSBs that define a region in the filesystem for file data or metadata
 - A single extent can be up to 8GB in length
- A file's inode lists the extents associated with that file
 - For very large files, the file's inode may have thousands of extents, or one very large extent. Usually something in between.
- Extents are also used for file and directory metadata when the information exceeds the space reserved for an inode

Unwritten Extents

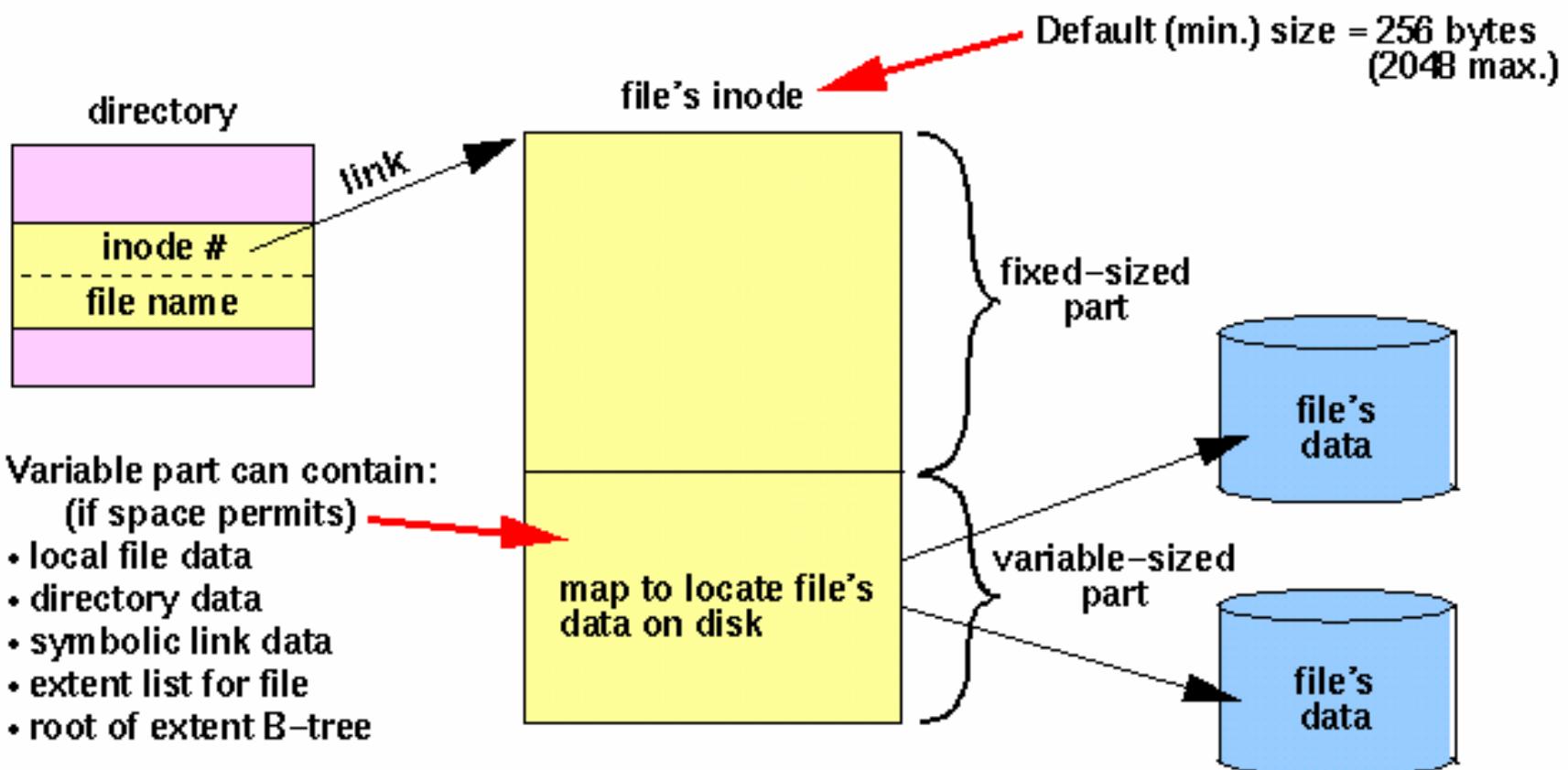
- An unwritten extent is an extent which has been marked as "not yet written" ondisk.
- Unwritten extents can be created by
 - Preallocating file space using (currently) XFS specific interfaces
 - Someday there may be a real kernel `fallocate(2)` syscall
 - Through direct IOs of specific (un)alignment.
- They are a security measure, to ensure allocated but not yet initialised space ondisk is not visible to arbitrary users
- `mkfs.xfs` option to enable or disable, on by default.
- Unwritten extents apply only to regular files.
- Once such an extent is written to, or partially written to, a transaction is issued to convert the written part into a regular written extent, and mark the remaining (up to 2) extents as unwritten.
- Use the `-p` option to `xfs_bmap` to view unwritten extents.

```
# xfs_io -f -c 'resvsp 0 10m' -c 'bmap -vp' /tmp/foo
```

Inodes

- XFS has three inode structures
- Ondisk inode
 - Used for storing the metadata for all files, directories and other file types
 - By default 256 bytes but can be up to 2KiB
- Linux inode
 - `bhv_vnode_t` has the Linux inode embedded in it
- XFS inode
 - `xfs_inode` contains the ondisk inode structure in memory
- The Linux and XFS inodes have different lifecycles, which can cause problems

Directory and File Inodes



Journal Log

- XFS Journal logs all metadata transactions
 - No record of data, only that the file size had changed
- Allows the filesystem to replay and recover the filesystem in seconds
 - No requirement to run fsck
- Log replay will apply filesystem and metadata changes that had been logged but may not have been applied to the filesystem when it went down
- The log may be located on a separate device

