SILICON GRAPHICS | The Source of Innovation and Discovery™

Silicon Graphics, Inc.

XFS Overview & Internals
09 - Internals

November 2006

XFS Architecture

- TODO: Incorporate Nathan’s stuff here

xfs_vnodeops

- VFS system call interfaces are mapped to xfs functions in `xfs_vnodeops`
  - fs/xfs/xfs_vnodeops.c

xfs_vnodeops {

- open, close, fid, read, write, sendfile, splice, fsync
  - file descriptors
- getattr, setattr
  - inode attributes - stat(2)
- attr_get, attr_set, attr_list, attr_remove
  - extended attributes
- access, lookup
  - inode permissions/existence
- create, remove, symlink, readlink
  - regular files, special files
- readdir, mkdir, rmdir, rmdir, link, rename
  - directories
- reclaim, release, inactive, iflush, bmap, flush_pages, flush_inval_pages, toss_pages
  - inode / page cache state and/or lifecycle

};

- todo: explain important vnodeops

xfsctl

- XFS specific system calls (`xfsctl()`) are dispatched by `xfs_ioctl()`
  - fs/xfs/xfs_fs.h
  - fs/xfs/linux-2.6/xfs_ioctl.c

- geometry, fscounts, [get|set]resblks, shutdown, freeze/thaw
  - filesystem level manipulation
- grow[fs|fslog|fsrt]
  - filesystem size (and maximum inode count) expansion
- [get|set]xflags, fs[get|set]xattr, fs[get|set]xattra, dioinfo
  - inode attribute information
  - direct I/O parameters (min/max/align)
- allocsp, freesp, resvsp, unresvsp
  - space allocation and/or preallocation
- bulkstat
  - many (sequential) inode's attributes – stat(2)
- xfsdump, quotacheck, dmapi
  - by-handle (open, fd-to-, path-to-, readlink, attrlist, attrmulti, ...)
  - manipulating inodes by “handles” (inum/igen/fsid)
- getbmap, getbmapa, swapext
  - inode data/attr fork extent information

- todo: explain important xfsctls

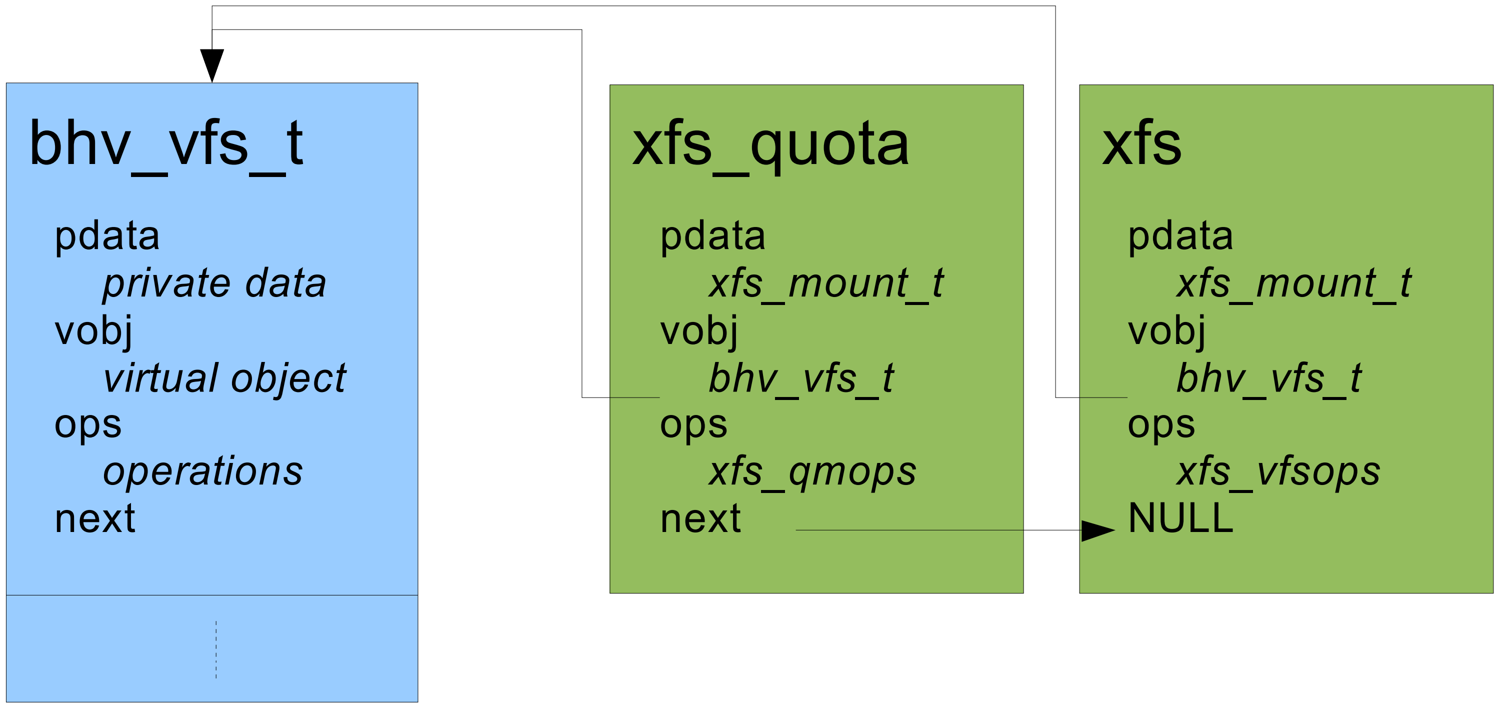
sysctl

- /usr/src/linux/Documentation/fs/xfs.txt

irix_symlink_mode	symlinks get mode 0777 by default
irix_sgid_inherit	sgid bit always inherited regardless of process gid
inherit_nosymlinks	Dont inherit symbolic links
restrict_chown	chown restricted to root
rotorstep	Number of files in AG before rotating to next group
probe_quota	Load kernel module on mount
probe_ioops	Load kernel module on mount
probe_dmapi	Load kernel moduel on mount
age_buffer_centisecs	Age of buffered data before flushing
xfssyncd_centisecs	How often xfssyncd runs
xfsbufd_centisecs	How often xfsbufd runs
inherit_noatime	Pass no accesstime tracking into file
inherit_nodump	Pass nodump flag into file
inherit_nosync	Pass nosync flag into file
error_level	Set XFS error handling level
panic_mask	Set XFS panic bits

dentry-state	Number of directory entries
	Number of unused entries
	Reclaim >secs when short on memory
	1 Calling shrink_dcache_pages
file-max	Maximum number of files system wide
file-nr	# files allocated
	Number of files in use
	Max number of files system wide
inode-state	Number of active inodes
	Number of free inode entries
	1 # > inode-max so prune inode list
inode-nr	same as inode-state first two vars

Behaviours

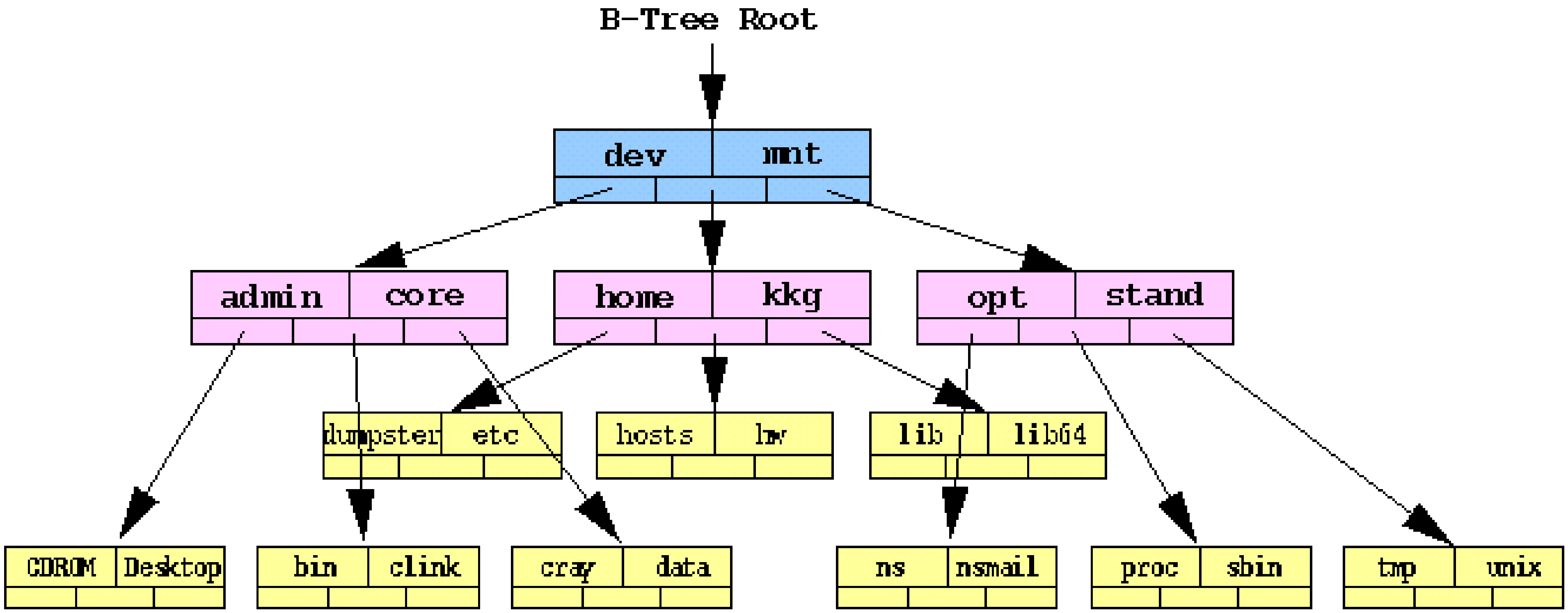


sgi®

Mount Path

- xfs_fs_fill_super
  - Allocate a bhv_vfs struct (*vfs_allocate*)
  - Setup initial behaviour module chain, for all bhv_modules (*bhv_insert_all_vfsops*)
  - Parse mount options (*bhv_vfs_parseargs*)
    - At the end of this we have the final behaviour chain – e.g. if quota is not in use, its removed itself from the chain *(bhv_remove_vfsops)*
  - Perform mount (*bhv_vfs_mount*)
    - For base XFS behaviour, we read the primary superblock, setup per-fs structures, does log recovery, etc.
    - For quota behaviour, we do the quotacheck and dquot recovery

B-Tree


B-Tree Root
dev
mnt
admin
core
home
kkg
opt
stand
dumpster
etc
hosts
hw
lib
lib64
CDROM
Desktop
bin
clink
cray
data
ns
nsmail
proc
sbin
tmp
unix

File and Directory Operations

Filename Lookup

Creating a new file

Allocating a new inode

Adding name to directory

Changing file attributes

Writing to a new file / Appending to an existing file

Reading from a file

Seek and write to create a hole

Read and write to a hole

Truncate a file

Space Allocation

- xfs_bmapi / xfs_bmap_alloc (the root of all evil!)
- Block MAP interface:
- access extent map for reading
- setup delayed allocation
- perform actual allocation
- convert unwritten extents to written extents
- Two space allocators
- Freespace B+Trees (“data”)
- xfs_bmap_btalloc
- Freespace bitmaps (“realtime”)
- xfs_bmap_rtalloc
- Other: stripe unit/width size/align, di_extsize

Memory Allocation

- Long been a source of problems on the Linux XFS port, it is much improved now, however.
- IRIX was very good at ensuring memory allocations succeeded, XFS written on IRIX... you do the math.
- Special process flag added into Linux XFS zone (slab) allocation routines that make the allocator aware of memory allocations from within a transaction.

Metadata Buffering

- xfs_buf.c and xfs_buf.h implements the XFS metadata buffer cache on Linux
  - Multi-page buffers
  - Buffer “pinning”
  - Several “private” buffer pointers
  - Locking, iodone semaphore for I/O waiters
  - Callbacks for: iodone, relse, pre-write
- In-core log buffers also implemented via xfs_buf_t and this causes some oddities in there – sub-buffer-sized I/Os, non-page-cache buffers, etc.
- Separate address_space from bdev

Metadata I/O Completion

- xfslogd/N (per-CPU daemon)
  - Threads that handle I/O completion work for iclog buffers
    - xlog_state_do_callbacks – runs multiple completions, depends on what was logged inside this iclog buffer)
  - and also metadata
    - xfs_buf_do_callbacks – typically, removing from AIL and freeing up buffer_item memory
- xfsdatad/N
  - will cover later, in the I/O path section
  - same sort of idea though

Delayed write buffers

- xfsbufd
  - kernel thread, one per filesystem device
  - walks the *xfs_buftarg_t* ("buffer target") hash table finding delayed write buffers
  - buffers timestamped when queued
  - can tweak the age at which unpinned and dirty metadata buffers will be considered for flushing
    - /proc/sys/fs/xfs/age_buffer_centisecs
  - tunable daemon wakeup interval
    - /proc/sys/fs/xfs/xfsbufd_centisecs

I/O Path

- read and write family of syscalls
  - both buffered and direct I/O
  - xfs_lrw.c
- Inode locking (i_mutex/iolock/ilock)
- DMAPI integration
- Delayed allocation
  - Initial write reserves space only, allocation at writeout time
- get_block_t interface
  - (inode, iblock, buffer_head, “create” flag)
- struct buffer_head
  - (b_state, b_blocknr, b_size, ...)

sync(2)

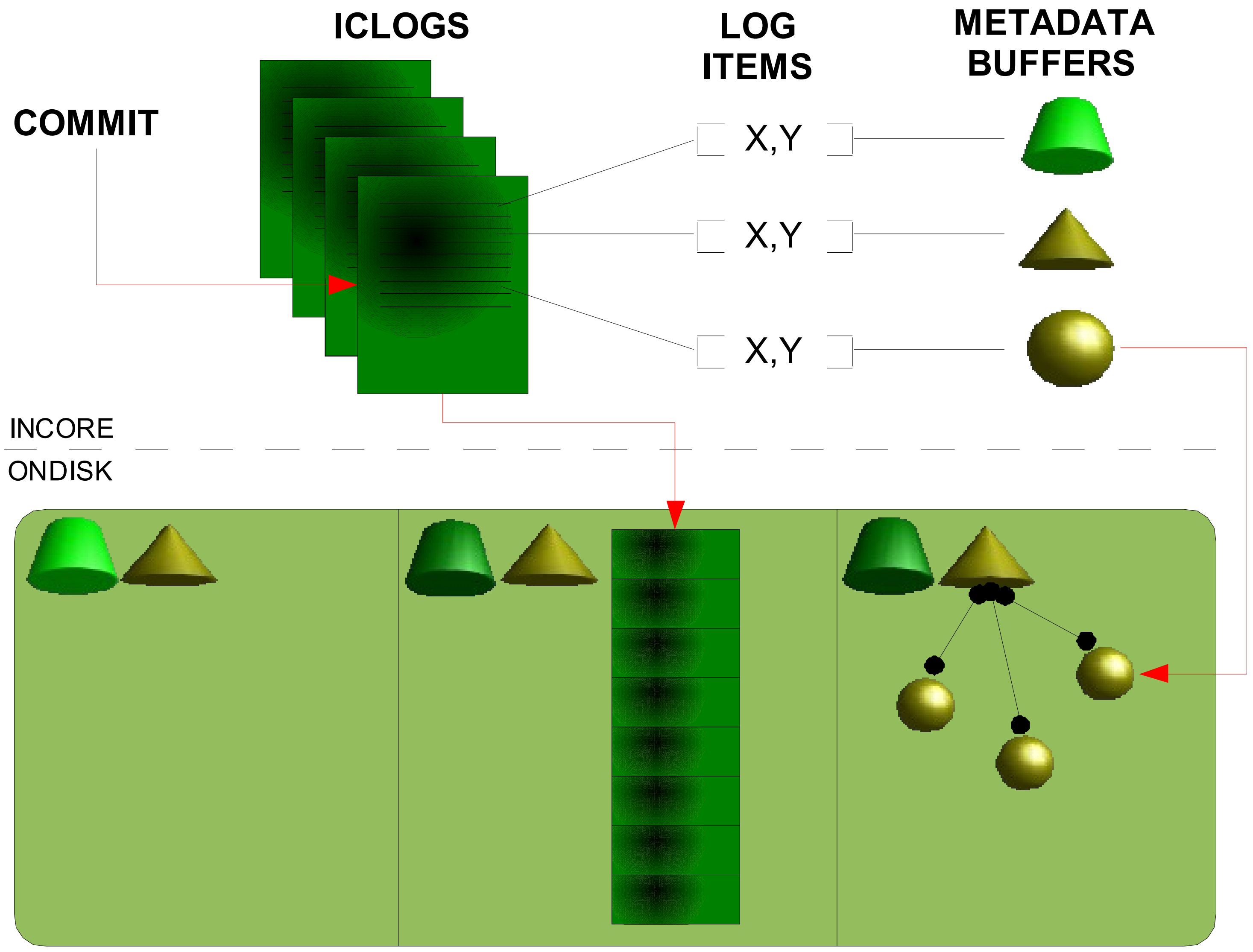
- XFS implements an optimisation to sync(2) of metadata:
  - XFS will only force the log out, such that any dirty metadata that is incore is written to the log only, the metadata itself is not necessarily written
  - This is safe, since all change is ondisk
  - File data is guaranteed too (even barriers)
- freeze/thaw, remount,ro and unmount do guarantee both log and metadata
- Applications like **grub** have been bitten in the past, but fixed nowadays

Data writeout

- Triggered by the VM subsystem
  - xfs_aops.c::xfs_vm_writepage(s)
  - xfs_aops.c::xfs_page_state_convert
- Page cache pages attached to inodes via a radix-tree (2.6)
  - inode->i_mapping and page->mapping
  - XFS does its own writeout, sort of (due to delayed allocation and unwritten extents)
- Walk through 2.6 writepage...
  - still use buffer_heads for per-fsbno state
  - xfs_ioend_t - goes direct-to-bio for actual write, with >1 page at a time

Transactions

- tp = xfs_trans_alloc(type);
- error = xfs_trans_reserve(tp, data, log, rt, ...);
- Then make changes, allocate space, free space, etc.
- Attach superblock/inode(s)/buffers/... to transaction, logging ranges within these objects, typically, e.g. via:
- xfs_trans_log_inode(tp, ip, XFS_ILOG_CORE);
- error = xfs_trans_commit(tp);


ICLOGS
LOG ITEMS
METADATA BUFFERS
COMMIT
X,Y
X,Y
X,Y
INCORE
ONDISK

sgi®