Storage Systems

NPTEL Course

Jan 2012

(Lecture 23)

K. Gopinath

Indian Institute of Science

Abstractions Used

Buffer related

- getblk: given a filesystem number and disk block number, get its locked buffer
- brelse: given a locked buffer, wakeup waiting procs and unlock it
- bread: read a given disk block into a buffer
- breada: bread + asynch. read ahead
- bwrite: write a given buffer to a disk block
- alloc: allocate a free disk block and return buffer using getblk
- free: free a disk block

Inode related

- iget: get a locked inode (doing bread if necessary) given inode number
- iput: release an inode; if ref count 0, writes dirty inode
- bmap: given inode and byte offset, returns disk block num and offset
- namei: given a path, get the locked inode
- ialloc: assign a new disk inode for a newly created file
- ifree: free an inode (link count 0)

Link (src, target)

- isrc = namei(src) (get inode for src)
- if too many links on file or linking dir without su, iput(isrc) (releases inode), ret err
- incr link count on inode, upd disk inode & unlock
- get parent inode (ptargetdir) of dir to contain new filename (uses namei)
- if new file exists, or src/target on diff fs, undo upd of inode and ret err
- create new dir entry in ptargetdir: new file name + isrc
- iput (ptargetdir) (release parent dir inode)
- iput(isrc) (release src file)

Write

- DAC/MAC?
- Locks? Range locks?
- Trigger? Freeze/thaw info?
- bmap
- Append/overwrite?
  - Alloc blocks?
  - NFS write?
  - Segmap? (kernel map of file?)
- Manage log space: Mem? Disk?
- Logging and flushing
- Manage visibility of curr write to other syscalls
- Manage VM allocs and buffers

General Issues

- Single threaded kernels: interleaved execution of getblk with interrupt handler or with brelse
  - May need to block interrupts in general
- Multi-threaded kernels: in addition, with other concurrent fs ops
- Avoid deadlocks at all costs
  - use lock ordering where possible!
    - Eg: in rename, have to lock source and target dir entries
    - (eg. zfs) lock dir with smallest object id first, or if it's a tie, the lexically first
  - where not possible to order, drop locks as necessary
  - marshall resources and be conservative to avoid deadlocks
- To freeze and thaw fs, need infrastructure
- With errors/triggers, have to store continuation and restore it