Storage Systems

NPTEL Course

Jan 2012

(Lecture 21)

K. Gopinath

Indian Institute of Science

FS Design

- Naming and Persistence
- Kernel level design common
  - Usually highly concurrent
  - Locking issues critical
    - eg. special handling for stat, mount/unmount
- User level (microkernel-based, some parallel fs, or higher level abstraction such as GFS)
  - Ops need not be designed to be concurrent in kernel
- Kernel level Design
  - Serves kernel components (eg. VM)
  - Serves appls (POSIX interface)
- Client of Device Drivers

FS

- Cannot operate on disk storage directly
- Metadata structures have “on-disk” and “in-memory” formats
  - Also have endian ness!
  - Disk formats usually more optimized
    - no locking info
  - Serialized before flushing time
- Clustering critical:
  - Eg: multiple inodes in one block (=> “false sharing”)
  - For block based disks
  - Also for current block based flash/SCM devices
- Caching critical; also prefetching
- Ordering flushes critical

Data Structures

- inode:
  - owner, access perms
  - file type (REG, DIR, FIFO, CHR, BLK, ...), file size
  - access times, #links, disk addrs for blocks in file
- incore inode: addl fields:
  - locked?, process-waiting?
  - dirty?, mount point?; reference count (# of opens),
  - ptrs to other incore inodes (free and hash q)
- superblock:
  - size of FS/inode list, dirty?
  - #free blocks/inodes, list/bitmap of free blocks/inodes, index of next free block/inode, locks for lists/bitmap

VFS

- To enable kernel to be independent of different types of filesystems, abstract interface VFS betw ker and fs
  - Also allows network file systems like NFS
  - Also pseudo filesystems like /proc, /tmpfs, ...
- Abstract class impl thru function pointers in C
  - Function pointers loaded at mount time
- However, kernel VFS infrastructure may not be same across OSes
- FUSE: help develop FS in userspace using VFS
  - FUSE kernel module intercepts call in VFS routine and upcalls to user code

Redundancy

- Critical structures replicated
  - Superblocks
  - Large scale systems: inodes also...
- May depend on volume managers
  - RAID
- To avoid corruption (eg. from hw faults)
  - Extensive self-consistency checks in code (just like in an OS)
  - eg: is buffer alloc safe? Invariant: alloc during syscall & free at end
    - Disk drive hw problem: cannot interrupt CPU: buf lost!
  - Best Effort Service
    - Soft errors (retry), hard errors
    - If error detected in spite of checks, try to move fs to some reasonable consistent state
    - Usually no non-starvation guarantees
- End-to-end checksums (across disks, HBAs, netw links) needed