Systemic problems in ZFS and ZFS/FreeBSD port

Andriy Gapon <avg@FreeBSD.org>

HybridCluster

KyivBSD, 2013

Overview

- Original ZFS problems
  - lots of kernel memory allocation
  - Solaris locking is not strict, especially with respect to memory allocations
- Problems for FreeBSD port
  - Very different VFS architecture
    - vnode life-cycle management
    - vnode locking, lock order
  - Integration with GEOM
    - lock order
  - Integration with VM
    - lock order
    - FreeBSD VM used to be less efficient when faced with a consumer such as ZFS

FreeBSD®

Overview

- Original ZFS problems
  - lots of kernel memory allocation
  - Solaris locking is not strict, especially with respect to memory allocations
- Problems for FreeBSD port
  - Very different VFS architecture
    - vnode life-cycle management
    - vnode locking, lock order
  - Integration with GEOM
    - lock order
  - Integration with VM
    - lock order
    - FreeBSD VM used to be less efficient when faced with a consumer such as ZFS

Spinlock

- spinlock
  - holds onto a CPU (spins)
  - current implementation disables interrupts
  - no operation that can switch a thread is allowed while holding a spinlock (no sleep, etc)
  - the only type of lock that can be acquired in fast interrupt handler

Non-sleepable locks

- mutex, rwlock, rmlock
  - if a lock is contended
    - inhibits the current thread, allows a different thread to run on the current CPU
    - waits for an owner thread to release
  - while holding a lock of this type
    - no sleeping is allowed
    - only locks of this type or spinlocks can be acquired
  - thus a thread that holds a lock either makes progress or waits on another process, and so on

Sleepable locks

- sx lock, lockmgr lock
  - anything is allowed while holding lock of this type
  - may sleep / wait

Sleepable vs Non-sleepable locks

- Advantages of non-sleepable locks
  - a thread could be blocked only by another thread
  - Easy to see thread and lock relationships
  - The dependencies must form a directed acyclic graph
  - witness(9)
- Sleepable locks
  - thread may wait for an event, not a thread
  - event can be anything
    - expiring timer
    - signal from another thread
    - event coming from hardware or another system
    - memory allocation with M_WAIT
  - impossible to accurately reason about dependencies
- Lock pools
  - N objects protected by M locks
  - witness is less effective

Solaris / ZFS locks

- Only spinlocks and sleepable locks
- mutex is sleepable and allows waiting memory allocation
- no witness
- no strict discipline in usage
- “magic dust” for memory allocation

ZFS specifics

- sx locks are used as a result of minimizing porting efforts
- almost all memory allocations are waiting
- lots of memory allocations while holding locks
  - not easy to untangle
- memory allocations in I/O path
  - buffer allocation for
    - compression
    - I/O aggregation
  - high deadlock potential when I/O is used to free memory
    - swapout for swap on zvol
    - pageout, affects everyone
  - deadlocks even in Solaris with KM_PUSHPAGE

High-level description of FreeBSD VFS

- fundamental concepts
  - use count, hold count
  - active, inactive, doomed, “larvae”
- VFS wants total control over vnodes
- VFS want a lot of control over filesystem behavior
- rich API
- almost everything that can be generalized tends to be generalized
- bias towards UFS/FFS as the only non-trivial native filesystem with long development history

Solaris VFS

- much smaller and simpler code
- much simpler vnode lifecycle at VFS level
  - but there are no miracles
- a lot of functionality has to be duplicated in filesystem code
  - but that gives filesystems more flexibility

Main porting problems

- FreeBSD vnode lock vs Solaris reference counting
  - in Solaris order of acquiring vnodes does not matter
  - deadlock potential
- vnode recycling
  - FreeBSD reserves a right to ask back an arbitrary vnode at an arbitrary moment
    - in practice a vnode in used is only reclaimed during forced unmount
  - In Solaris a filesystem recycles its vnodes
    - no references remain[1]
    - forced unmount
- FreeBSD VFS does not provide support for rollback-like operations
  - Solaris VFS does not need that and it is completely delegated to filesystems

[1] Solaris namecache acts similarly to FreeBSD free / inactive vnode list

Pipe dreams

- Convert ZFS locks to native non-sleepable locks
- Avoid waiting memory allocations
  - Especially while holding internal locks
  - Keep reserves for I/O initiated by pagedaemon
- Decouple znode lifecycle from vnode lifecycle
  - treat znode like a Solaris vnode and provide Solaris lifecycle management
  - znodes would have internal reference count
  - gfs[2] can be re-implemented in the same way or re-done like FreeBSD devfs

[2] used to implement virtual entries under .zfs

Thank you!

- Thank you for listening!
- Questions?