OpenAFS and Large Volumes

Mark Vitale <mvitale@sinenomine.net>

European AFS and Kerberos Conference
28 March 2014

objective / overview

- We will explore some of the current challenges and solutions for using OpenAFS with very large volumes.
- “How big is a piece of string?”

case study

A large university stores research project data on a single OpenAFS volume:

- millions of files and directories, almost 1 terabyte total
- content-addressable directory hierarchy
  - e.g. one directory per person “query_subject_nnnnn”
  - AFS files-per-directory limits forced awkward workarounds
- original volume okay, but unable to make a usable copy
  - ‘vos move/copy/dump/restore’ would run for many hours then fail
  - triggered salvage which would run for many more hours, then either fail or produce damaged data
  - workaround: tar/untar “WAY out of band”

outline / agenda

- big data considerations
- internals
- addressing the issues
- futures
- questions and answers

big data considerations

- architectural limits
- implementation limits
- server limits (disk, memory)
- operational limits (time, SLA)
- configuration changes
- performance issues (scalability)
- bugs – increased exposure of infrequently exercised code paths

salvager

SINE NOMINE ASSOCIATES

- one of those programs (like restore) that is not used very often, but when you do you really hope it works!
  - "I do not always corrupt my volumes. But when I do, I use salvager."
- looks for errors and inconsistencies in AFS volume data and metadata, and makes a best effort to repair them.

namei partition format

- **/vicepXX** partition
  - **Vnnnnnnnnn.vol** files (VolumeDiskHeader)
  - **AFSIDat** directory
    - 1st level volume directories (hashed lower bits of RW volid)
      - volume directories (hashed RW volid)
        - 1st level inode directories (hashed high bits of vnode number)
          - 2nd level inode directories (hashed next 9 bits)
            - inode files (more hashing)
        - **special** directory
          - volume info – 1 per volume in the VG (RW, RO, BK)
          - large vnode index – 1 per volume in the VG
          - small vnode index – 1 per volume in the VG
          - link index – 1 per VG
- architectural limit of namei format
  - 26-bit vnode number = 67,108,863 max vnodes per volume – half small, half large
- data and metadata redundancies

addressing the issues

SINE NOMINE ASSOCIATES

- files-per-directory limit
  - detailed explanation in Mike Meffie's EAKC 2012 presentation: http://conferences.inf.ed.ac.uk/eakc2012/slides/eakc-2012-dir-defrag.pdf
  - workaround: implement site-specific nesting of user directories to skirt the AFS limits
  - workaround: use shorter file and directory names when possible

addressing the issues

SINE NOMINE ASSOCIATES

- errors and corruption during volume operations (copy/move/dump/restore)
  - “Can AFS make a volume so big he cannot move it?”
  - **root cause:** incorrect (signed) type for a vnode number led to a 31/32 bit overflow when calculating an offset into a vnode index.
  - patch submitted to fix the specific issue (gerrit 10447)
  - ongoing work-in-progress to audit and fix:
    - 31/32 bit (signed/unsigned) type errors
    - type-dirty macros (index offsets) – convert to inline static functions
    - other datatype error cleanup

addressing the issues

SINE NOMINE ASSOCIATES

- salvager assertion fail while salvaging a large volume
  - exposed by vos move bugs (previous slide)
  - root cause: SalvageIndex had an incorrect (signed) type for a vnode number; this caused the vnode index offset logic to overflow at 2^31 and pass a negative file offset to IH_WRITE
  - patch tested
  - work in progress (previously mentioned)

addressing the issues

SINE NOMINE ASSOCIATES

- mkdir may crash dafileserver
  - bug 1: allocating a new vnode overflows the index logic at 2^31 due to incorrect type; issues “addled bitmap” and goes to rarely-traversed error recovery code
    - work-in-progress (previously mentioned)
  - bug 2: error recovery mishandles volume reference counting and crashes on a failed assert
    - patch tested; will be submitted to gerrit soon

addressing the issues

SINE NOMINE ASSOCIATES

- salvager resource requirements
  - uses “invisible” (unlinked) temp work files
    - e.g. salvage.inodes – contains info on each inode
    - default location is on the partition itself
    - may grow quite large for large volumes, leading to unexpected disk space exhaustion during salvage
    - recommend using –tmpdir parm to put these elsewhere
  - uses large memory-resident tables
    - “vnode essence” tables that “distill” the vnode index metadata
    - may result in a visit from the Linux OOM-killer

addressing the issues

SINE NOMINE ASSOCIATES

- configuration changes
  - just creating the test volumes exposed some problems in my cell config:
    - cache manger: memcache/disk cache; separate partition; separate device; adequate stat caches (to prevent vcache invalidation and VLRU thrashing)
    - fileserver: adequate callbacks (to prevent callback GSS)
  - first accesses exposed more perf considerations
    - cold CM caches
    - cold fileserver caches
    - ls coloring causing extra FetchStatus calls
    - CM prefetch activity

addressing the issues

- performance & scalability
  - vos move/copy/dump/forward profiling
  - salvager dir buffer optimizations
  - fileserver housekeeping optimizations

futures

- Finish current audit/bug hunt/cleanup project
  - including submitting all work upstream to gerrit
- Death to salvager?
  - Why do we even have a salvager?
    - to repair (after the fact) damage caused by:
      - hardware software failures
      - AFS design shortcomings (non-logging, etc?)
      - AFS bugs
  - DAFS improved some pain points in the salvager:
    - greatly reduced salvage related outages
    - salvage on demand, with no outage for other volumes
    - salvage in parallel
- Extended directory objects?

Questions?