OpenAFS Unix Cache Manager Performance

SINE NOMINE

Mark Vitale <mvitale@sinenomine.net> AFS and Kerberos Best Practices Workshop 20 August 2015



objectives

- Understand the performance characteristics of the OpenAFS Unix Cache Manager:
 - frequent causes of performance problems
 - configuration options
 - performance measurement & troubleshooting
- Have a better answer the next time your phone rings and you hear this from the other end of the line...



"Why is AFS slow?"



round up the usual suspects

- the network: packet drops?
- the DB servers: loss of quorum?
- the volume servers: failed volume release?
- the fileservers: threads waiting? callback space exhausted?
- the KDC: okay, now you're just guessing....
- They're all fine what else could it be?



the cache manager?

- The OpenAFS Cache Manager is quite complex, and does much more than a typical light-weight client.
- It maintains lots of different kinds of caches, not just one.
- One Cache Manager is shared by all AFS users and applications on a given host system.
- Sometimes the Cache Manager is a "server" too.



common pain triggers

- large parallel workloads against AFS content
 - parallel builds; research analysis runs; LSF batch jobs
- live RW content (one writer, many readers)
 - message queues; logs
- s.l.o.o.o.o.o.o.o.o.o.o.w writer
- multiple NAT traversals
 - virtual machines; working from home



potential causes

- file contention (writes are exclusive)
- internal lock contention AFS_GLOCK, other global locks
- callback storms
- somebody else's network



cache manager configuration

- lots of options if you want to tweak
- the default auto-tuning is almost always fine if you don't
- pick your cache size and the rest is calculated based on that
- for "server" type loads, look at –stats and volumes



-fakestat

- -fakestat (and –fakestat-all) are performance optimizations to improve interactive user experience while browsing /afs with:
 - OS X Finder
 - other Linux GUI file navigators
 - Unix ls -1 (or ls with coloring on)
- as the name implies, they do this by "faking" replies to stat() calls in order to avoid contacting the fileserver.
- often causes performance problems with large cache manager "server-like" workloads.



server preferences

- internal CM "rankings" for choosing a server when there's multiple choice
- not configurable via afsd options!
- defaults are often fine
- fs getserverprefs [-vl] [-numeric]
- sometimes you may want to "steer" with
 - *#* fs setserverprefs



cache mgr troubleshooting (basic)

- rxdebug
- cmdebug
- xstat_cm_test
- rxping, rxtraceroute



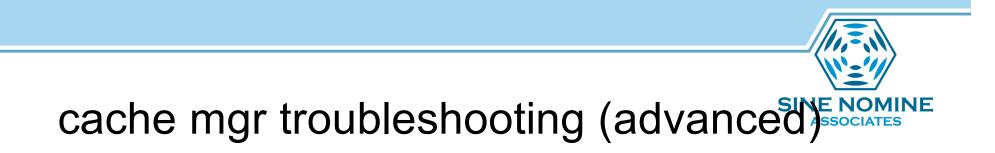
cmdebug

- cmdebug <cm_host_or_IP> [options]
- by default (no options), displays current locks
- can *degrade* performance if you have lots of –stat entries
 - the vcache/callback scanner converges on $O(n^2)$
- config query options are always safe:
 - cache, -addrs, -cellservdb



xstat_cm_test

- xstat_cm_test <cm_host_or_ip> -onceonly collID <n>
 - n=0 internal routine call counts
 - n=1 -not currently implemented-
 - n=2 config settings, performance counters
 - fileserver RPC counts & response times



- fstrace
 - mostly for developers
- Linux CM only:
 - echo t > /proc/sysrq-trigger
 - slightly disruptive
 - echo c > /proc/sysrq-trigger
 - very disruptive



futures

- global lock mitigation
- make the housekeepers better housemates
- "watch this space"



Questions?