AFS at Intel

Travis Broughton
Agenda

Intel’s Engineering Environment
Things AFS Does well
How Intel uses AFS
How not to use AFS
Management Tools
Intel’s Engineering Environment

Learned about AFS in 1991
First deployed AFS in Intel’s Israel design center in 1992
Grew to a peak of 30 cells in 2001
Briefly considered DCE/DFS migration in 1998 (the first time AFS was scheduled to go away...
~95% NFS, ~5% AFS
~20 AFS cells managed by ~10 regional organizations
AFS used for CAD and /usr/local applications, global data sharing for projects, secure access to data
NFS used for everything else, gives higher performance in most cases
Wide range of client platforms, OSs, etc
Cell Topology Considerations

Number of sites/campuses/buildings to support
Distance (latency) between sites
Max # of replicas needed for a volume
Trust

... As a result, Intel has many cells
Things AFS Does Well

Security

- Uses Kerberos, doesn’t have to trust client
- Uses ACLs, better granularity

Performance for frequently-used files

- e.g. /usr/local/bin/perl

High availability for RO data

Storage virtualization

Global, delegated namespace
AFS Usage at Intel:
Global Data Sharing

Optimal use of compute resources
• Batch jobs launched from site x may land at site y, depending on demand

Optimal use of headcount resources
• A project based at site x may “borrow” idle headcount from site y without relocation

Optimal license sharing
• A project based at site x may borrow idle software licenses (assuming contract allows “WAN” licensing)

Efficient IP reuse
• A project based at site x may require access to the most recent version of another project being developed at site y

Storage virtualization and load balancing
• Many servers – can migrate data to balance load and do maintenance during working hours
AFS Usage at Intel: Other Applications

x-site tool consistency
- Before rsync was widely deployed and SSH-tunneled, used AFS namespace to keep tools in sync

@sys simplifies multiplatform support
- Environment variables, automounter macros are reasonable workarounds

“@cell” link at top-level of AFS simplifies namespace
- In each cell, @cell points to the local cell
- Mirrored data in multiple cells can be accessed through the same path (fs wscell expansion would also work)

/usr/local, CAD tool storage
- Cache manager outperforms NFS
- Replication provides many levels of fault-tolerance
Things AFS Doesn’t Do Well

Performance on seldom-used files
High availability for RW data
Scalability with SMP systems
Integration with OS
File/volume size limitations
When NOT to Use AFS

CVS repositories
  • Remote $CVSROOT using SSH seems to work better

rsync

Any other tool that would potentially thrash the cache...
Client cache is better than nothing, but shared “edge” cache may be better

- Mirroring w/ rsync accomplishes this for RO data
- Client disk is very cheap, shared (fileserver) disk is fairly cheap, WAN bandwidth is still costly (and latency can rarely be reduced)
OpenAFS at Intel

Initially used contrib’d AFS 3.3 port for Linux

Adopted IBM/Transarc port when it became available

Migrated to OpenAFS when kernel churn became too frequent

Openafs-devel very responsive to bug submissions

- Number of bug submissions (from Intel) tapering off – client has become much more stable
Management Tools

Data age indicators
- Per-volume view only
- 11pm (local) nightly cron job to collect volume access statistics
  idle++ if accesses==0, else idle=0

Mountpoint database
- /usr/afs/bin/salvager –showmounts on all fileservers
- Find root.afs volume, traverse mountpoints to build tree

MountpointDB audit
- Find any volume names not listed MpDB
- Find unused read-only replicas (mounted under RW)

Samba integration
- Smbklog

“Storage on Demand”
- Delegates volume creation (primarily for scratch space) to users, with automated reclaim
Management Tools

Recovery of PTS groups
• Cause – someone confuses “pts del” and “pts rem”
• Initial fix – create a new cell, restore pts db, use pts exa to get list of users
• Easier fix – wrap pts to log pts del, capture state of group before deleting
• Even better fix – do a nightly text dump of your PTS DB

Mass deletion of volumes
• Cause – someone does “rm –rf” equivalent in the wrong place (most recent case was a botched rsync)
• Initial fix – lots of vos dump .backup/.readonly | vos restore
  Disks fill up, etc
• Other fixes – watch size of volumes, and alert if some threshold change is exceeded
  Throw fileserver into debug mode, capture IP address doing the damage and lock it down
Management Tools

Watch for ‘calls waiting for a thread’
Routing loops can trigger problems
True load-based meltdowns can be diagnosed

• Send signal to fileserver to toggle debug mode
• Collect logs for some period of time (minutes)
• Analyze logs to locate most frequently used vnodes
• Convert vnum to inum
• Use find to locate busiest volume and files/directories being accessed
• Sometimes requires moving the busy volume elsewhere to complete diagnosis
Management Tools

Keep fileserver machines identical if possible

- Easier maintenance

Keep a hot spare fileserver around and online

- Configure as a fileserver in local cell to host busy volumes
- Configure as a DB server in its own cell for DB recovery

“Splitting” a volume is somewhat tedious

- Best to plan directory/volume layout ahead of time, but it can be changed if necessary
Questions?