Goldman Sachs

Leveraging AFS Storage Systems to Ease Global Software Deployment

Tracy J. Di Marco White
Vice President

Engineering Division

Summary

The ability to reliably distribute rapidly changing content to hundreds of thousands of client systems globally using significantly less network bandwidth and comparatively small amounts of local disk storage drives our use of AFS.

What is AFS?

Features

Network Based Distributed File System

Location independent

Global File Namespace

Consistent client layout

Distributed object service

Locally cached

Read only volume replication

Layered security

Goldman Sachs

What is AFS?

AuriStorFS

Backward compatible

Significant speed

Multi-layer security providing:

- Fine grained access control
  - File system objects
  - Volumes
- Volume based security polices
- File server based security policies

Uniform audit streams for all server requests

AES-256 encryption

AuriStor clients handle out of order packets better

Significant scaling improvements

Magic Mount

Goldman Sachs

Why do we use AFS?

We use AFS to provide software distribution at scale to hundreds of thousands of servers, deployed throughout private and public cloud, managed by separate units independent from the AFS engineering team.

AFS allows distribution of tens of terabytes of software to these clients using less than 10GB of local space on each with close to the speed of local disk, supported by 3-6 AFS file servers per cell

Why do we use AFS?

Features

Atomic sub-minute software deployment

Upwards of a hundred thousand client nodes

Provide somewhat under 30TB of software to clients

Client side cache uses 10GB

Client side cache allows minimal network use

Commodity hardware or cloud hosted virtual machines

Ease of management at scale

Global name space

Why do we use AFS?

Read Only Replication

Read write volumes replicated into read only volumes

Process is called a release

Copy on write on the server, read write to read only clone on read write server effectively instantaneous

Releases incremental when possible

Volume version changes deploy in seconds for small volumes and small change sets

Cache consistency guarantees are provided by server notifications when volume releases occur

What does our AFS infrastructure look like?

and why?

Tens of AFS cells, each with 3 to 6 object (file) servers

Each cell can support tens of thousands of clients

Some cells have AFS aware distribution of software

Some have AFS unaware distribution of software

Some have both.

Each cell has its own AFS-aware and/or AFS-unaware writer on a client host.

Our current AFS usage is overwhelmingly read only

Two software deployment systems?

Similarities

Deployment of software packages with possible interdependencies

- Content providers publish packages
- Primary sites store published packages
- Hub sites in each cell store subscribed packages, requested from primary sites
- Hosts and host groups subscribe to packages from the hub sites
- Packages provided via AFS, NFS, and local host copies
- Path volumes contain mount points for package volumes
- Supports Linux and Windows

Two software deployment systems?

Differences

AFS Unaware Deployment

- Part of one particular SDLC
- Provides a consistent set of software, evolving synchronously
- Distributes ~300 software packages with >6 versions to hundreds of thousands of hosts
- Scheduled, regular releases for all related software
- One of the firm’s more mature offerings

Two software deployment systems?

Differences

AFS Aware Deployment

- Strategic choice for new software deployment
- Supports multiple SDLCs
- Provides declared dependency resolution
- Distributes 384k active packages to hundreds of thousands of hosts
- Individual package distribution
- Subscribed packages are always available via the same path

Two software deployment systems?

Who are the clients?

Both software deployment systems are on a majority of client hosts

Read only clients

- Software developers
- Service providers
- Software deployment
- Internally and externally developed software users
- Infrastructure management

- Everyone at the firm using modern Linux is using one or both deployment systems
- Everything at the firm leverages software deployed using one or both deployment systems

Goldman Sachs

AFS unaware deployment of software

How is this architected?

File system hierarchy is pre-existing, with few path volumes and few large software volumes

Volumes are mutable, changing with each new package update

Each AFS cell has its own writer for the software deployment system

Software on the AFS servers monitor for pause in writing

- Perform releases when a volume requires update and is quiescent
- Care taken to prevent release of inconsistent volumes
- No communication between deployment writer and AFS servers

Currently ~500GB

AFS unaware deployment of software

What problems do we see?

Replication and writing methods means changes to volume take up additional space on the server until release

AFS servers do not know whether the deployment writer paused for a short or a long time

All OpenAFS clients and older AuriStorFS clients require a 10 minute pause between releases of the same volume

Larger volumes require longer replication times between servers

Resiliency requirements include three copies of each volume

Replication takes one or more offline at a time until release is complete

AFS unaware deployment of software

How is this being re-architected?

The deployment writer is becoming AFS aware.

- Deployment writer releases volumes
- Individual software package and version each go into their own volume
  - Deployment writer creates volumes for each package/version
  - Reducing size of large volumes as package versions iterate into their own volume
  - A few path volumes may still see repeated changes due to volume mount points needing release
- Growth is being planned to ~5TB

AFS aware deployment of software

How is this architected?

The software distribution writer writes each package version written into a separate read write volume it creates

Software package volumes are immutable once written

Performs releases when package writing is complete

Path volumes change with each new package version

Multiple path volumes spread the need to release the path volume

Volumes written are small, released only once for packages, released as needed for path volumes

Symlink tree created from dedicated path into AFS hierarchy

AFS aware deployment of software

What problems do we see?

Older AFS clients require a 10 minute pause between releases of the same volume

- Path volumes see rapid changes due to volume mount points needing release
  - Example: A single change to software available on multiple platforms, each in its own volume
  - Order of volume release critically important, path volume must be released after the package volume

Failure of the AFS server hosting the read write path volumes prevents further releases of software

AFS aware deployment of software

How is this being re-architected?

Using the AuriStorFS feature Magic Mount, treat AFS as an object store

- /afs/.@mount/cell/volume is treated as a normal AFS mount point
- Avoids multiple directory lookups that existed due to the paths in use for the deployment hierarchy and AFS
- Symlink tree created from dedicated path to Magic Mount volume location

AFS aware deployment of software

How is this being re-architected?

- Removes the need for path volumes
  - All volumes are immutable
    - No callbacks need to be broken
    - Release ordering becomes a non-issue
  - Not taking up space in the client cache

AFS aware deployment of software

How is this being re-architected?

- 50% of package deliveries complete in one minute, previously three minutes
- 90% of package deliveries complete in four minutes, previously thirteen minutes
- Must upgrade to current AuriStorFS client to take advantage of Magic Mount enabled software deployment

What issues are seen?

How are they mitigated?

Magic Mount

- Made visible a software anti-pattern, use of ../.. for path lookups, libraries, other files
  - AuriStor added negative volume name-to-ID lookup caching to client and server to mitigate impact
  - Separated AFS file servers from volume location servers in larger cells
  - Work with client customers to replace use of ../.. to be more targeted

What issues are seen?

How are they mitigated?

AFS file servers are mostly cattle, not pets, hosts holding RW volumes are special

- Magic Mount removes most of the need for the RW volume for AFS aware deployment
  - Older clients cannot use Magic Mount
- Magic Mount not yet an answer for AFS unaware deployment

What issues are seen?

How are they mitigated?

Growth of working set (client software utilization set) beyond size of local disk cache

- Increasing size of local disk cache implies a host rebuild
- Reduces available local disk on existing deployments

Goldman Sachs

What's next?

AFS Unaware Deployment possibilities

- Creating more path volumes to mitigate against single points of failure
  - Avoiding path volumes still a better choice
- Moving toward single package/version volumes, immutable
  - Create directly using Magic Mount
  - Consider symlink farm
  - Immutable volumes may be able to be created directly as read only
    - Removes need for read write site

What's next?

AFS Unaware Deployment

- Mutable volumes still necessary
  - Limit mutable volumes to very few, very small
- Coalesce single path volume changes to mitigate 10 minute pause necessary for older clients

What’s next?

AFS Aware Deployment

AFS Aware Deployment possibilities

- May create read only volumes directly
  - Deployment writer can create replicas on any server

What’s next?

Wish list

- AuriStor servers backed by Ceph or equivalent for increased resiliency
- Ability to mark immutable volumes as such, reducing need for server promise (callback) expiration
- Compressed data transfer to clients – decrease network usage
- Faster, smaller deltas

Summary

The ability to reliably distribute rapidly changing content to hundreds of thousands of client systems globally using significantly less network bandwidth and comparatively small amounts of local disk storage drives our use of AFS.

Thank you for your attention today!