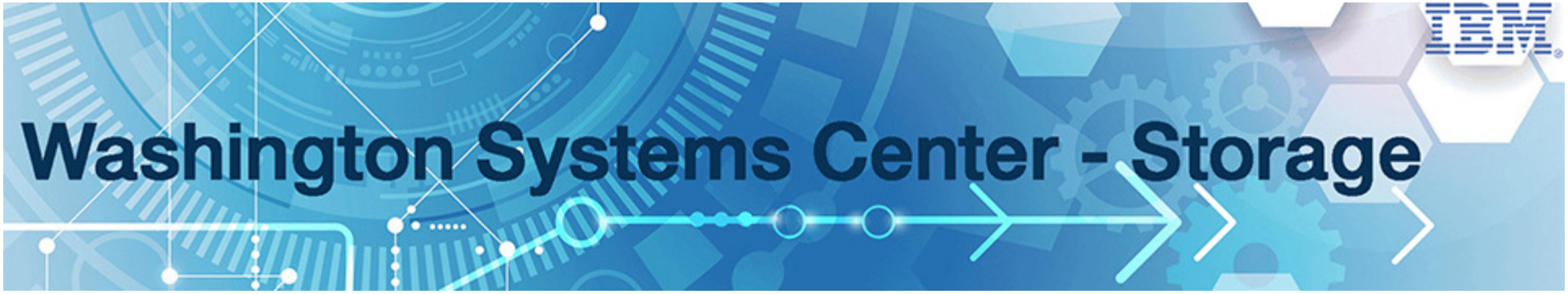


Deep Dive into Spectrum Scale AFM

R. Lindsay Todd, PhD
Ken Hill
Gilbert Nevarez
Connie Rice

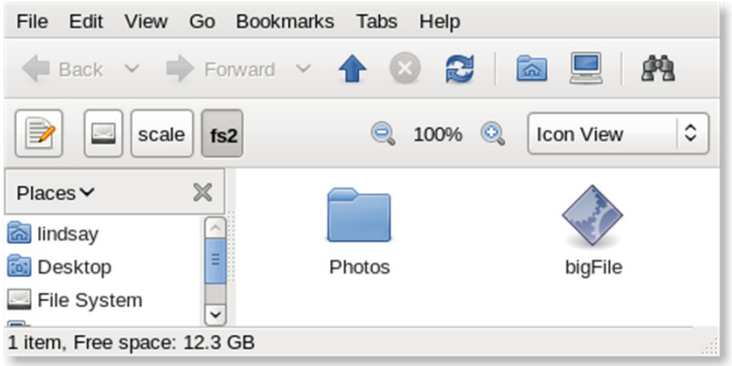
Agenda

- Introduction to, motivation for, and explanation of Active File Management (AFM)
- Use cases for AFM
- Deep dive into Active File Management concepts
- Working with AFM
- Working with AFM-DR

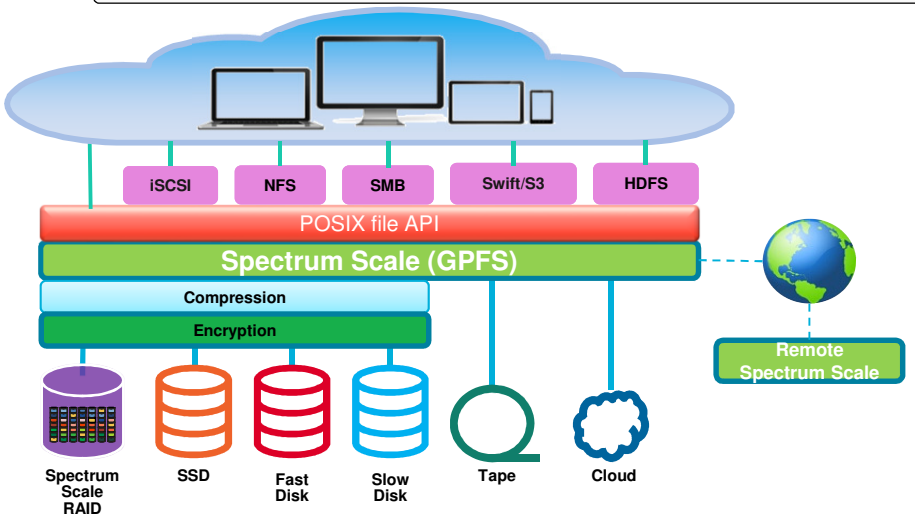
Spectrum Scale: What *is* it?

Spectrum Scale is a system-level ***software application***:

- implementing a **highly scalable distributed *parallel* POSIX file system**,
- able to run on systems under **Linux, AIX, and Windows**
- **builds upon any block storage** (from local disk to SAN-attached, not necessarily from IBM),
- providing **advanced data management features** built in to it that go well beyond the capabilities of traditional file systems
- with **features layered on top** of it to make that storage accessible through NFS, SMB, Object, and iSCSI protocols.




```
$ pwd
/scale/fs2
$ ls -l
total 20608
-rw-r--r--. 1 root root 21098464 Jan 13 15:57 bigFile
drwxr-xr-x. 2 root root     4096 Jan 13 15:56 Photos
$
```



Active File Management: What *is* it?

Active File Management (AFM) lets Spectrum Scale extend over geographic distances

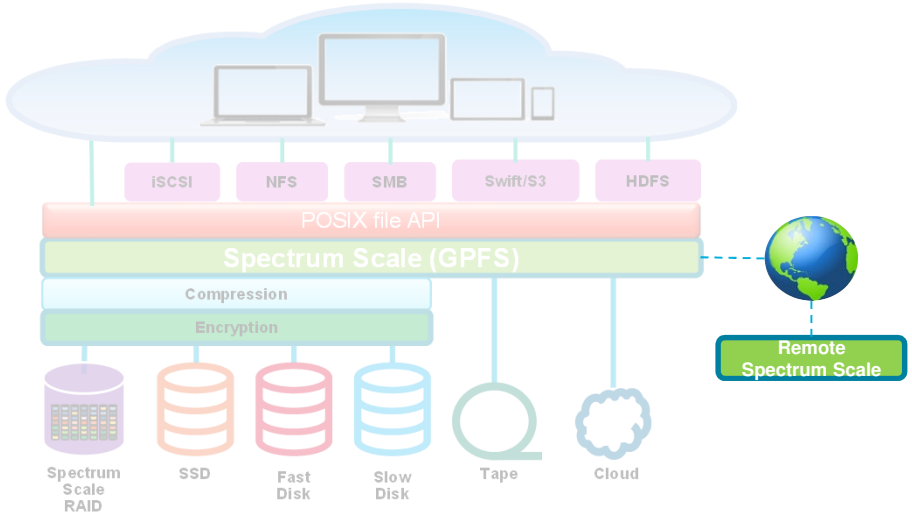
- Tolerates unreliable, high-latency networks (like a WAN).
- Caches copies of data from a remote file system into the local Spectrum Scale cluster.
- Cached files have the same read and write performance as other local files.

Enables efficient data access to collaborators and resources around the world

- Unifies heterogenous remote storage

Asynchronous DR (AFM-DR) is a special case of AFM

- Bidirectional awareness for failover and failback with data integrity
- Recovery Point Objectives (RPO) for application consistency



Active File Management – Motivation

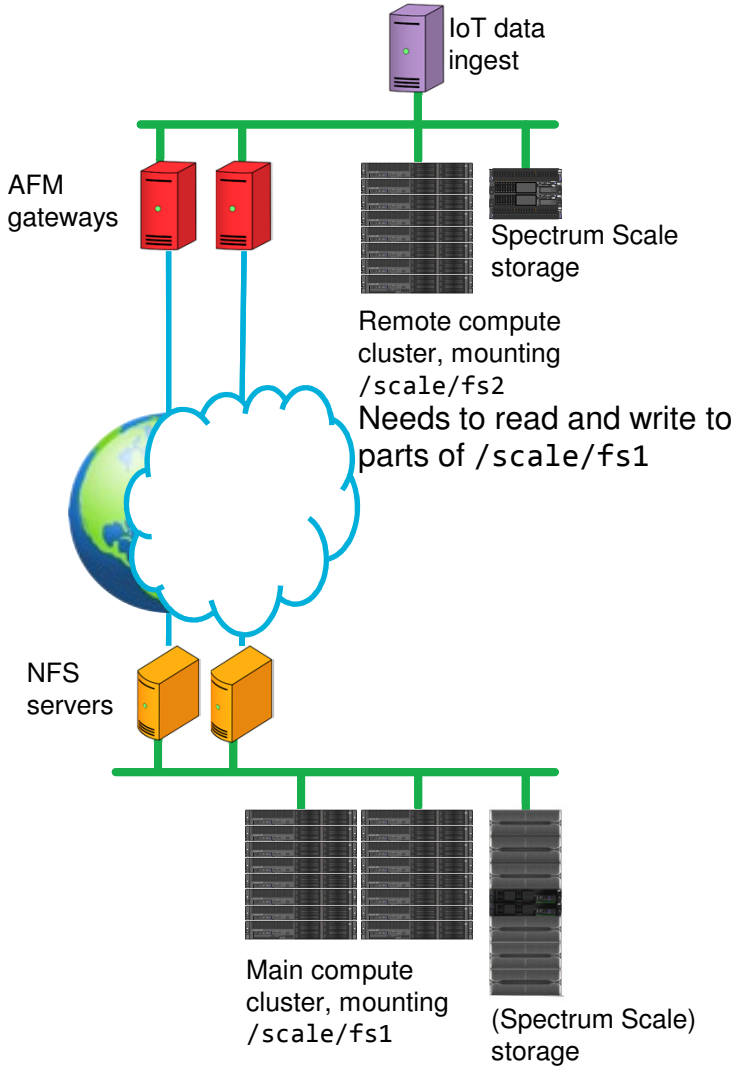
A main data center has large (Spectrum Scale) file system, `/scale/fs1`, and a main compute / application cluster.

A remote site is doing its own computation. It might be remote because of special data acquisition or ingestion needs. Data is stored locally in `/scale/fs2`.

But the remote site also needs data from `/scale/fs1`, and needs the main compute cluster to work on data it has ingested (ideally stored in `/scale/fs1`).

We can't afford the latency of accessing data stored across the world, and we need the performance of Spectrum Scale.

Active File Management (AFM) lets us set up filesets in the remote cluster as caches of parts of `/scale/fs1`, using NFS transport.



Active File Management – Conceptual overview

AFM *Cache filesets* cache data from a data source manifested as target or *Home filesets*, using several modes:

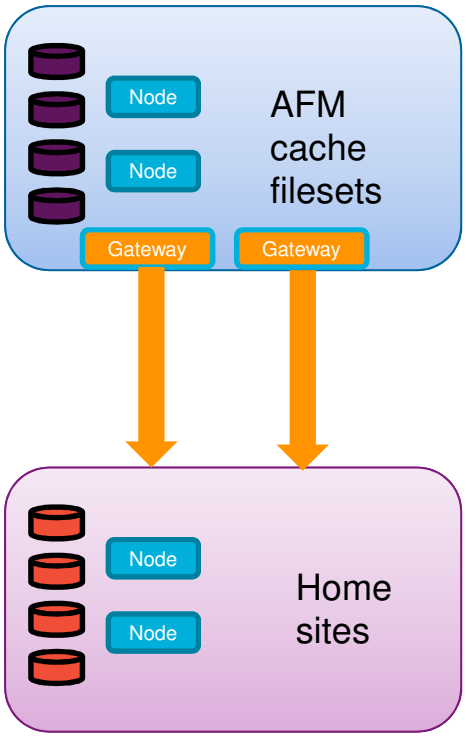
- Read-only: Data in Cache fileset is read-only (RO)
- Local update: Data from Home is RO, new files may be created, modified
- Single writer: Home fileset may be modified only by one Cache fileset
- Independent writer: Allow multiple Cache filesets to write to Home

Cache filesets transfer data using *gateway nodes*. The cache can be used even during times when WAN link is unavailable.

Home sites (often filesets), available through either Spectrum Scale NSD protocol or NFS, are the actual data source. The home site does not need to be in Spectrum Scale.

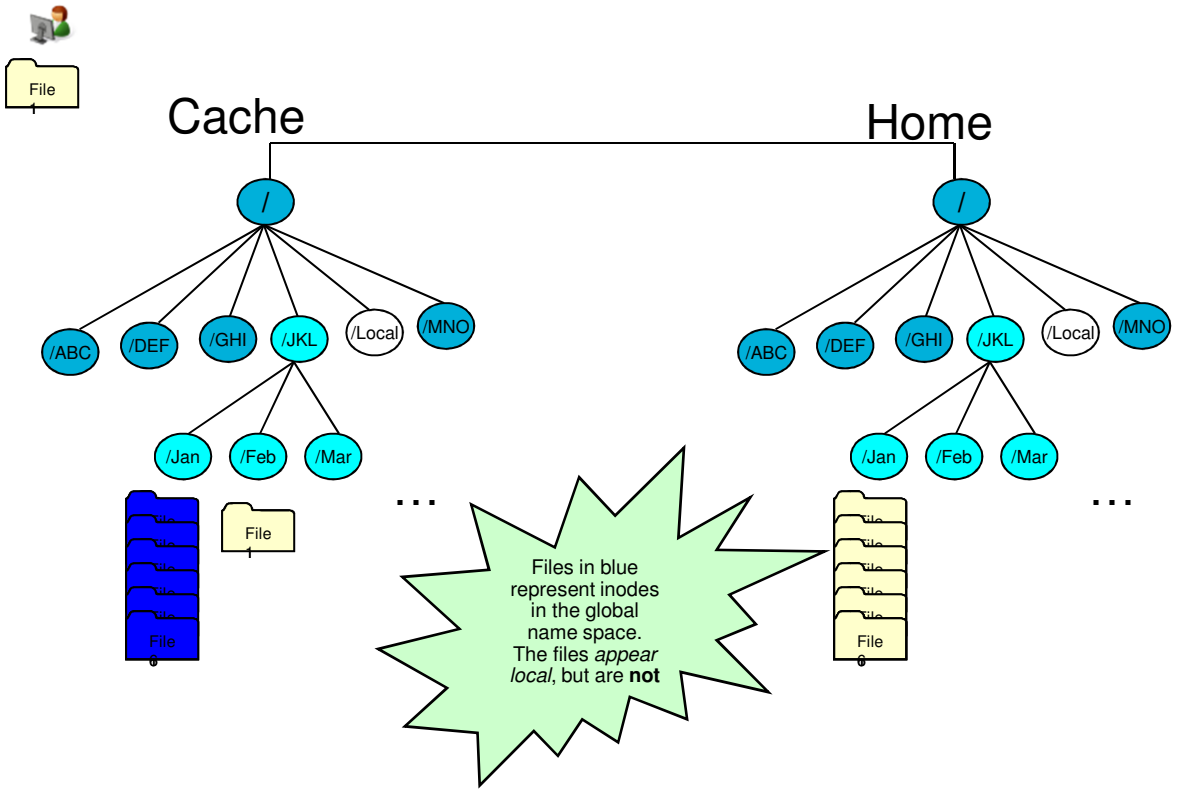
Updates from cache to home are *asynchronous*. ***Revalidations*** of cached items are on demand. Update granularity is the file system **block**.

Caching is on demand, *not automatic replication from home to cache*.



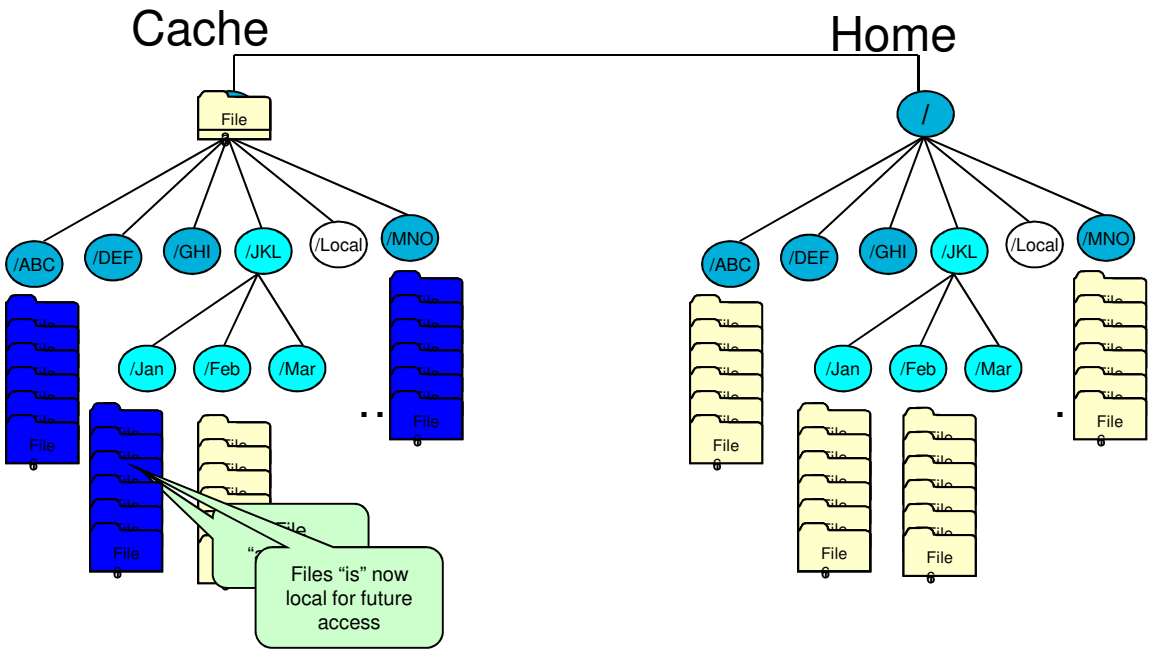
AFM: Write a File

Cache user writes a file on February 1st that is copied when possible to Home Site



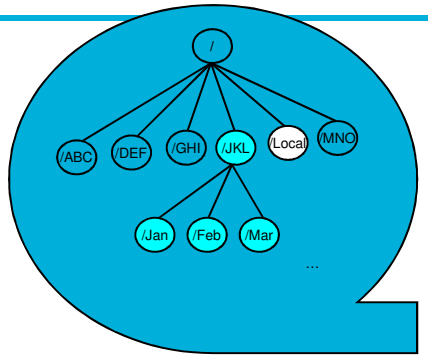
Read a File from “Home”

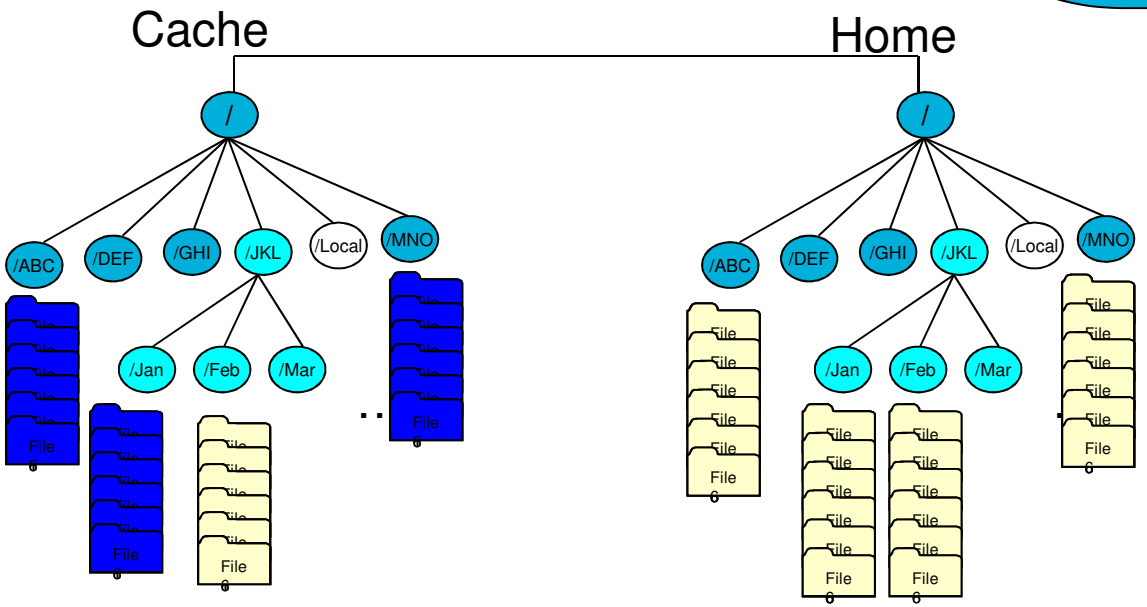
Cache user “double clicks” on /JKL/Jan/file3 in his directory that is not local



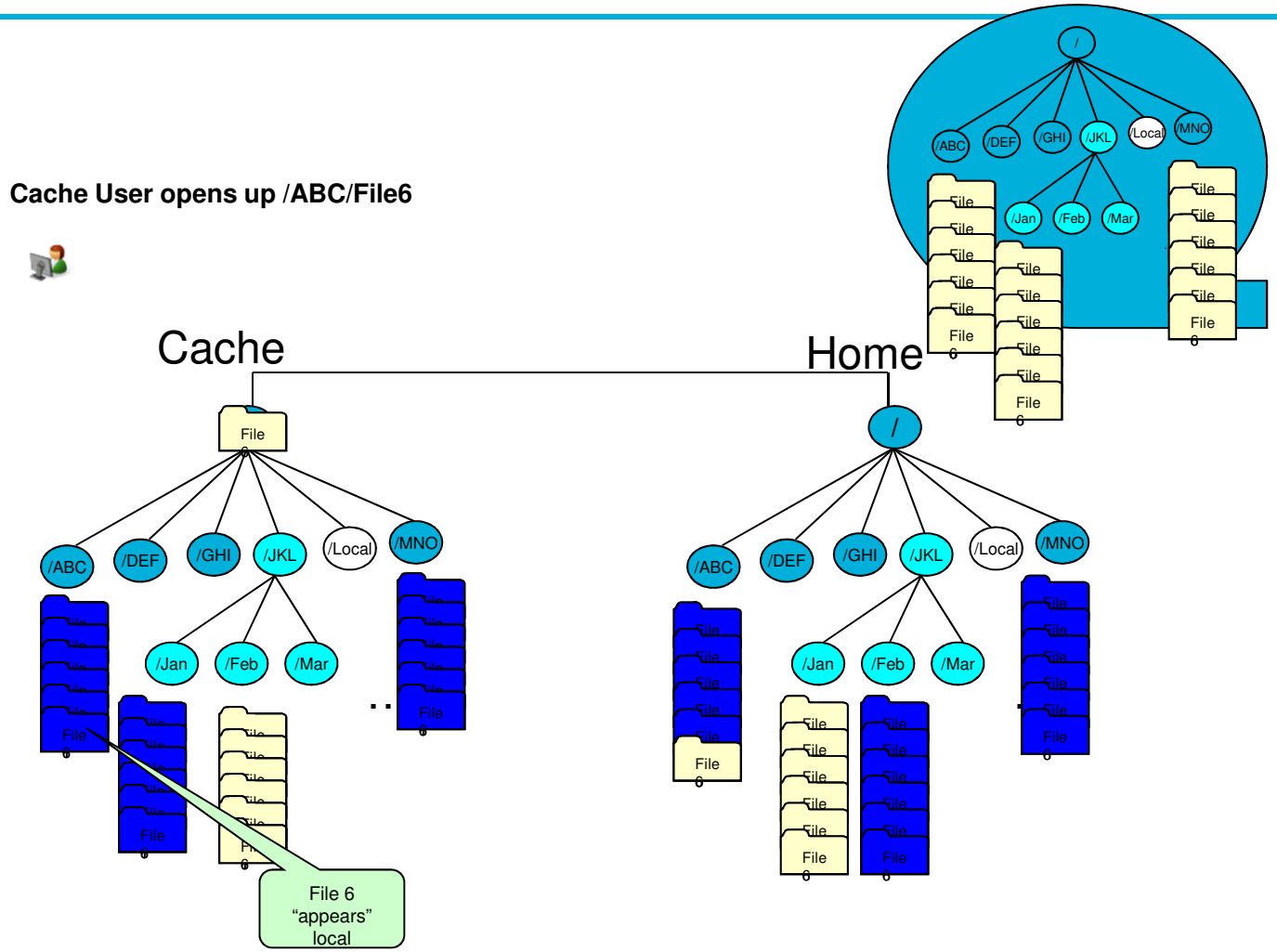
Archive to tape

Archive data 30 days old to tape by policy





Demand File Pull



AFM use cases

yes, this really is being used…

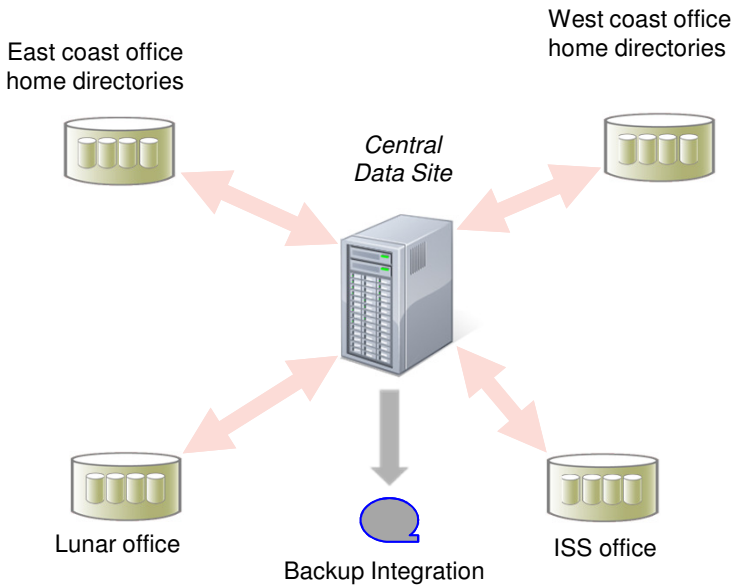
Central data storage with AFM caches at branch offices

Central data center contains “home” filesets.

- Backups, archives, etc. are handled at this single central site.

Branch offices have local caches.

- Caches can be small but fast.
- AFM masks much of the latency and unreliability of WAN links.
- Central site contains the “master copy” of all data, so loss of a cache is not itself catastrophic.



Accelerating NFS / Migrating from NFS using AFM

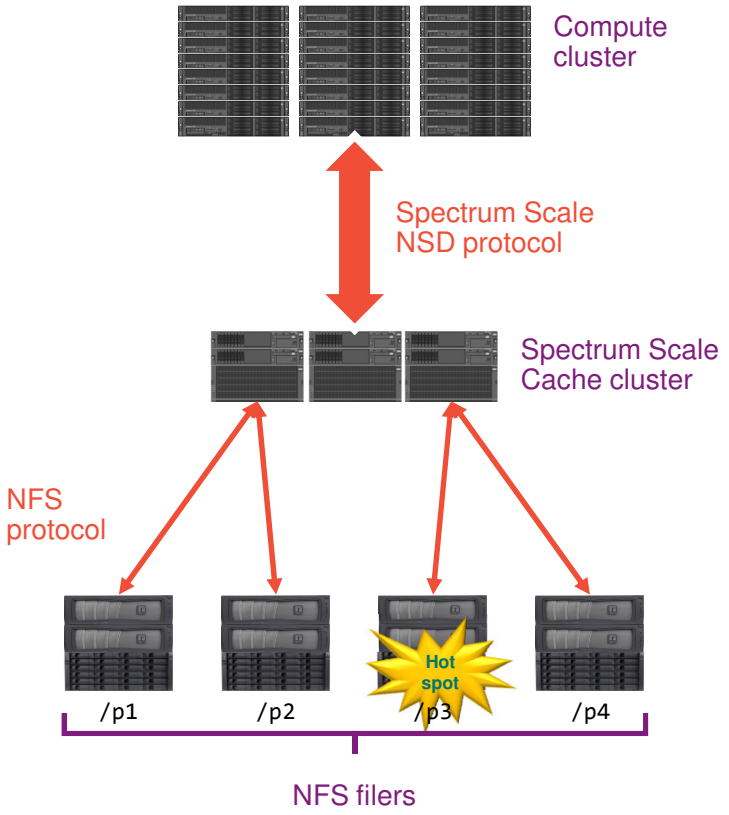
Interposing Spectrum Scale between NFS filers and the compute cluster can:

- Provide parallel read/write performance to files on the NFS filers
- Give consistent performance to files, hiding hot spots
- Unify separate name spaces into a single file system

This can also provide a **migration strategy** from NFS filers to Spectrum Scale, minimizing down time:

- Point applications to cache, and immediately they benefit from Spectrum Scale without losing access to data.
- Configure cache to never expire data.
- **Prefetch** all data, to also transfer what is not organically cached by running applications.

This use case makes NFS data available through Spectrum Scale. But it does not magically imbue the filers with Spectrum Scale's capabilities! E.g., filers will not suddenly be able to use Transparent Cloud Tiering…



AFM-based burst buffer

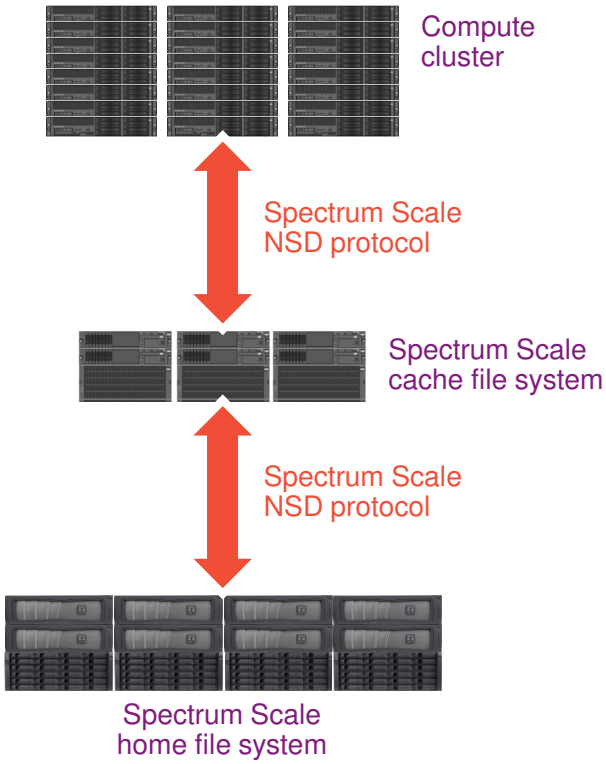
Fast storage (Flash? NVMeOF?) is configured into a separate Spectrum Scale file system, with cache filesets – both file systems can reside in the same cluster.

AFM relationships are established between filesets in main storage and cache filesets – usually using NSD protocol for transfer.

Client compute nodes access files through the cache.

- All new files created in the cache.
- Hot data remains in cache.
- Colder data, when warmed, moves into cache (evicting colder data) – this happens upon demand (unlike ILM, which requires the policy engine to run).

AFM ensures all cached data is *also* written to the home file system as soon as possible.



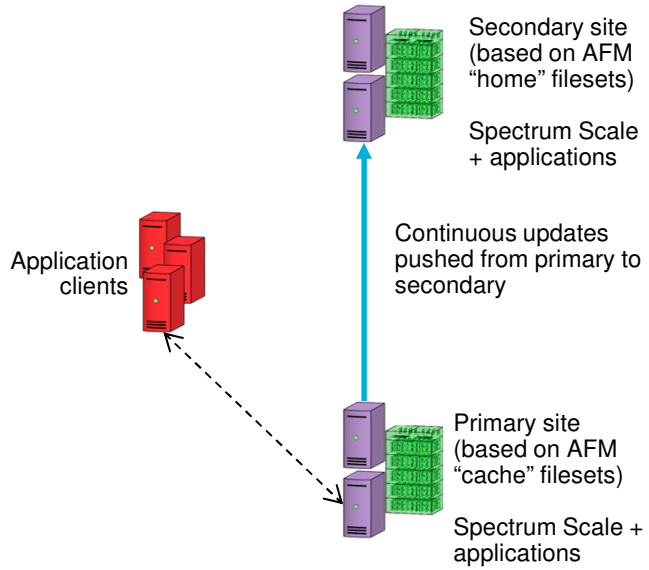
AFM Asynchronous DR

Primary site is consists of AFM cache filesets, targeting home filesets at the secondary sites.

Applications normally work with the caches (primary filesets). AFM ensures file updates are propogated (“replicated”) to the secondary site.

On failover, applications can be pointed to the secondary site.

Upon failback, the primary filesets can recover changes that were made to the secondary, and applications can be pointed back to the primary site.



How is AFM being used today?

Major cloud services provider

- Distributes data read-only to global sites.

Life sciences

- Sequencer data pushed to remote site

Several major banks

- AFM-DR to push data backups

Mechanical engineering software company

- Uses AFM for satellite offices

US Ivy League university

- Used AFM to migrate data onto new system

AFM concepts

“Things should be as simple as possible. But not simpler.” – A. Einstein

AFM gateways

One or more Spectrum Scale nodes are designated to be ***AFM gateway*** nodes.

Each cache fileset is automatically assigned to one of the gateway nodes to be its ***primary gateway***.

- Other nodes accessing the fileset (including other gateway nodes) communicate with the primary gateway to maintain cached data.
  - This communication, and gateway node responsiveness, puts some overhead on AFM, 1.2x to 4x.
- The primary gateway may enlist other gateways to assist with parallel data transfers.
- If a primary gateway fails, other gateway nodes take over its filesets, recover AFM, and upon return of the original primary gateway, transfer control of the filesets and work queues back to it.

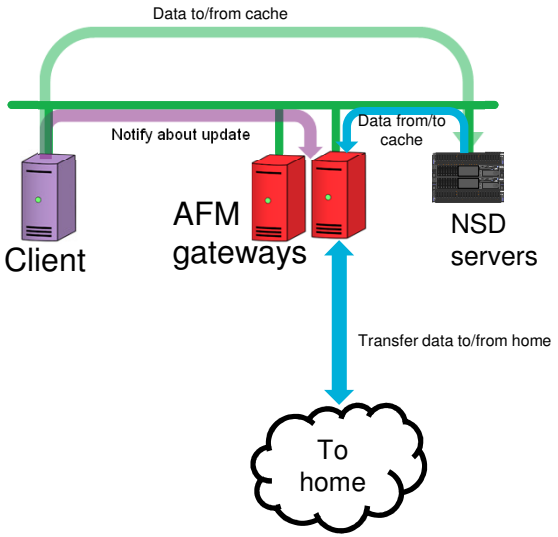
Plan about 10 filesets per gateway node.

- Tested with 100 AFM filesets, 100 million files / fileset

A good gateway:

- 128GB RAM
- Disk space in /var/mmfs/afm – figure:
  (filesets/gateway)*(files in fileset)*255 bytes
- Avoid other functions that might contend for local disk, e.g., an AFM gateway is not an ideal quorum node.
- Do not put the gateway function on a CES (protocol) node.
- In general, AFM gateway nodes should be dedicated to this purpose.

AFM gateway nodes require a Spectrum Scale “server” license.



AFM mode: Read-only caching

Read-only (RO) caching mode:

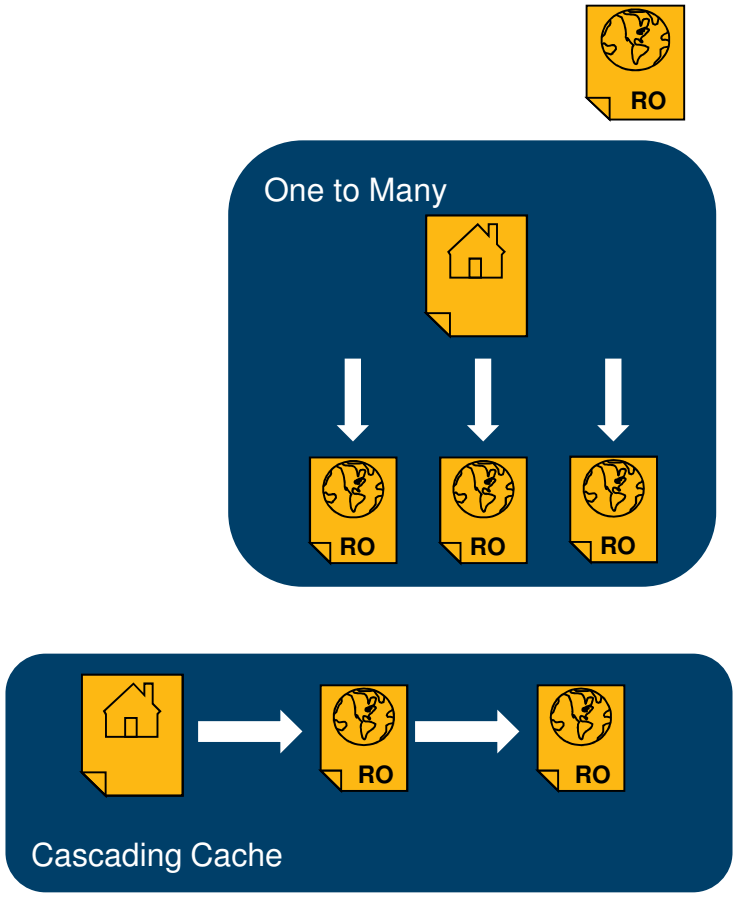
- Data exists on the home and one or more cache sites
- Nothing can write into the cache – it is literally *read-only*.

Data is copied to the cache on-demand.

- File metadata caching: Listing the contents of a directory copies the file metadata information into the cache
- Data caching: Opening a file copies the data to the cache
- The cache can also be populated using a prefetch with a list of files.

Caching behavior:

- Cached files are *revalidated on demand*, when an interval has passed, to pick up changes from the home.
- Many to one: A single home can have multiple RO caches.
- Cascading caches: An RO cache can be the home to another RO cache.
- Optional LRU cleaning of cache: If the cache fills and allows expiration, least-recently used data is removed.



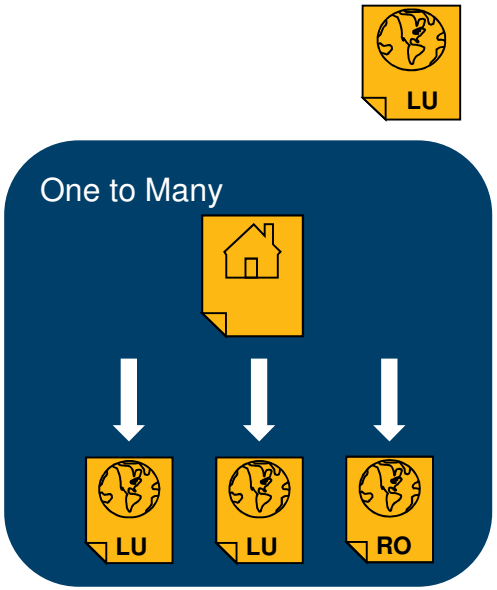
AFM mode: Local updates caching

Local updates (LU) caching mode:

- Data exists on the home and one or more cache sites

An LU cache largely behaves like an RO cache, except:

- *Data can be written into the cache* – but it will not be propagated back to the home.
- Once a file is modified in the cache, it will no longer be revalidated.
- NB. Once a directory has been updated, it will not be revalidated either! This can lead to surprises.
  - Subdirectories will still be revalidated, unless they too are modified.
- Multiple RO and LU caches can point to the same home.



AFM mode: Single-writer

Single writer (SW) caching mode:

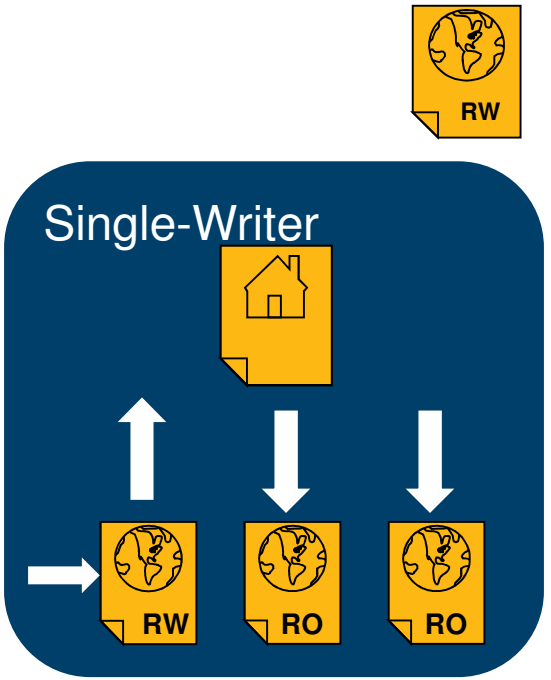
- When caching is first established, preexisting data will be accessible through the cache.
- Data may be written to a cache, and files may be changed.

Cache updates are pushed to the home asynchronously.

- There is no revalidation of files and directories in the home.
  - Thus, be sure that nothing else will change the contents of the home!
- You can also have multiple RO and LU caches connected to the home, but only a single SW cache.

Peer snapshots

- The `mmpsnap` command create a snapshot of the cache, and a corresponding snapshot of the home.
- You probably want the home to be an independent snapshot (otherwise the peer snapshot will be of the entire file system).



AFM mode: Independent Writer

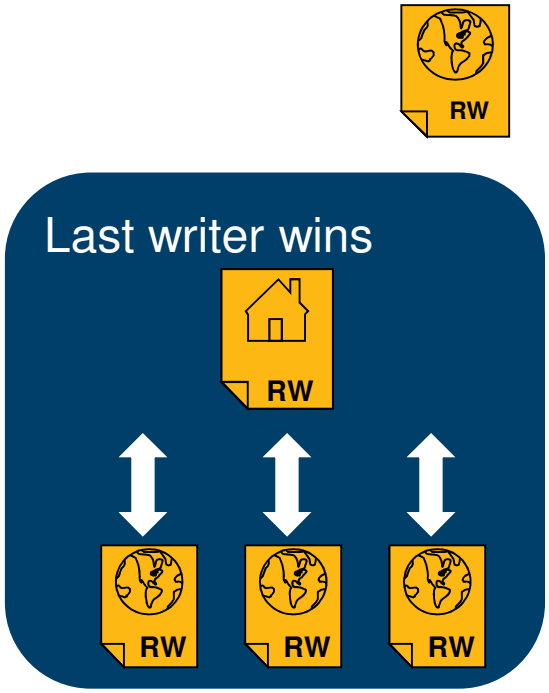
Independent writer (IW) caching mode:

- You may have multiple IW caches of the home.
  - You may also have RO and LU caches of the same home, but no SW caches.
- All IW caches may write data, which will propagate to the home.
- Local processes can change the home directly.
- Cached data is revalidated on demand.

Conflict resolution:

- There is no locking or ordering of updates between different caches and the home, so there is no defined conflict resolution.
- Effectively, it is: **The last writer wins**

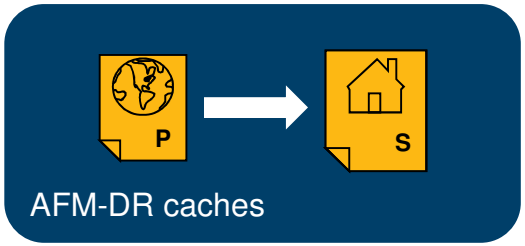
Peer snapshots are not supported with IW caches.



AFM mode: Primary / Secondary

AFM-DR caching modes:

- Primary (DRP) fileset behaves mostly like an SW cache fileset.
- Secondary (DRS) fileset behaves mostly like an independent fileset, but is read-only.
- On failover, the secondary fileset becomes a writable acting primary.
- On failback, the primary fileset is updated from the secondary.
- It is possible to completely reverse roles.
- Both filesets may be (re)created from independent filesets with existing data.



The AFM-DR fileset modes are only available with Spectrum Scale Advanced Edition or Data Management Edition.

- The Standard and Data Access editions support all other AFM functions.

Converting between AFM fileset cache modes

Cache mode	Allowed to switch to…
Read Only	Local Update, Single Writer, disabled
Local Update	disabled
Single Writer	Read Only, Local Update, Primary, disabled
Independent Writer	Read Only, Local Update, Single Writer, Primary, disabled
Primary	Secondary, disabled
Secondary	Primary, disabled
Non-caching independent fileset	Primary, Secondary

All AFM filesets must be independent filesets.

An AFM cache mode can not be added later to a fileset (except AFM-DR modes).

A fileset must be unlinked when its mode is changed.

Single Writer and Independent Write filesets may not have their cache mode changed if there are still any updates pending.

Controlling cache eviction

Cache eviction ensures the cache doesn't fill up.

Cache eviction of file blocks automatically starts when the fileset's soft block quota limit is reached.

- This may be changed using the `afmEnableAutoEviction` fileset parameter.
- Dirty files will not be evicted.
- The soft inode quota has no particular significance to AFM.

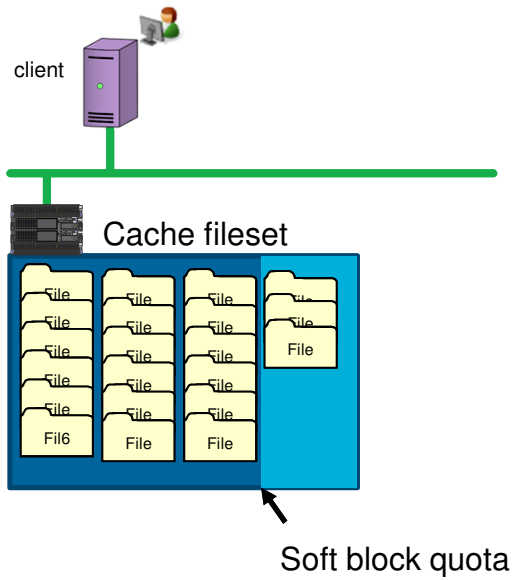
Files may evicted with more control using the `mmafmctl evict` command.

- A list of files, perhaps created using the policy engine, can be provided.
- Different sorting policies and thresholds control what files will be evicted.
- Only non-dirty files are evicted.

Eviction removes only the file data from the cache.

- The metadata (not a stub) remains.
- Accessing the file will bring its contents back into the cache.

1. Files written to cache
2. Soft block quota exceeded
3. Least recently used files evicted



Controlling cache revalidation

Cache revalidation ensures the data in the cache is fresh.

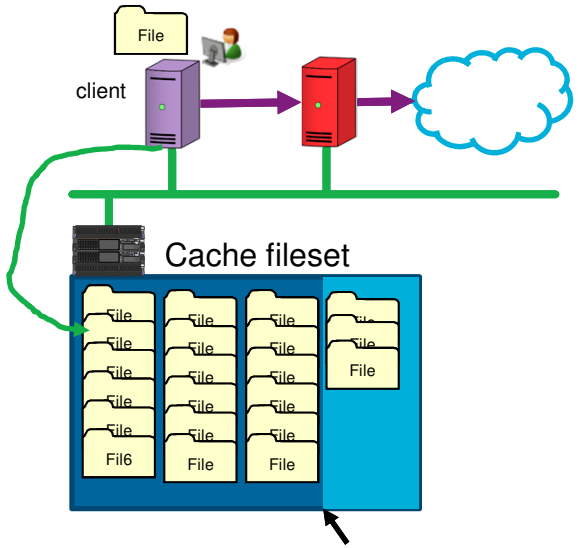
When files and directories are accessed, the cached data is periodically checked to see if it current, relative to the cache, and updated if necessary.

Several configuration variables control how frequently revalidation is performed:

- `afmFileLookupRefreshInterval`
- `afmDirLookupRefreshInterval`
- `afmFileOpenRefreshInterval`
- `afmDirOpenRefreshInterval`
- `afmRevalOpWaitTimeout` (cluster-wide setting)

Revalidation is when-needed (not automatically), per node (not fileset). Only RO, LU, and IW filesets periodically revalidate. SW filesets revalidate on first use only.

1. Request to open a file
2. Node checks when last revalidated; it is longer than `afmFileOpenRefreshInterval`
3. Node requests gateway to revalidate file
4. File is opened



Transport between home and cache

Every cache fileset has a ***target***, specifying both the location of the home and the protocol used to communicate.

Protocol choices:

1. Spectrum Scale NSD protocol – gateways access home as parallel clients.
2. NFS v3 protocol – primary gateway uses NFS to transfer from home.
3. A "map" associating NFS v3 servers to gateway nodes (allows parallel transport with multiple NFS servers)
   - Each gateway node is assigned to an NFS server.
     - Be sure to include an assignment for each gateway node, lest unexpected behavior arise.
   - An NFS server may be assigned to more than one gateway node, in which case:
     - Only one gateway will handle any read task
     - Large write tasks will be split amongst gateway nodes

Use `mmafmconfig` to enable a Spectrum Scale home to support AFM caches with sparse files and extended attributes, even when NFS v3 is the transport. (The `mmafmconfig` command is also used to configure maps.)

The `afmEnableNFSSec` setting enables Kerberos with NFS v3 for encrypted transport. NSD transport can also be encrypted.

How to choose

With a stable, low-latency network, choose the **NSD** protocol (generally will perform better).

For less reliable or high latency networks (typical WAN), choose **NFS** protocol (tolerates failures, packet drops, while better isolating cache from the home cluster).

How transported

Reads and revalidations are synchronous. Files can be partially cached (`afmPrefetchThreshold` controls)

Writes are asynchronous. AFM normally transfers only dirty blocks. The `afmAsyncDelay` setting indicates how often this happens.

Writing to a file caches the entire file, unless only appending to end.

Gateway and Network Failures

Failure of gateway

When a gateway node fails:

- Another gateway takes over primary gateway function.
- Since the in-memory queue was lost, ***recovery*** is needed.
  - Recovery begins when the fileset becomes active, typically first access.
  - The `afmRecoveryStart` and `afmRecoveryEnd` callbacks can be used to monitor when recovery is needed.
- The ILM policy engine is internally used to scan the cache fileset to build the ***priority queue*** of update requests.
- Only SW and IW caches need recovery.
- The `afmMaxParallelRecoveries` variable controls how many filesets may be recovered simultaneously. (0 means do everything in parallel.)

Failure of network

The `afmDisconnectTimeout` tells the primary gateway how long to wait before declaring the home to be disconnected, triggering the `afmHomeDisconnected` callback.

For NSD transport, if the home fileset is deadlocked, revalidations may hang. (Revalidation can be disabled while home is unavailable.)

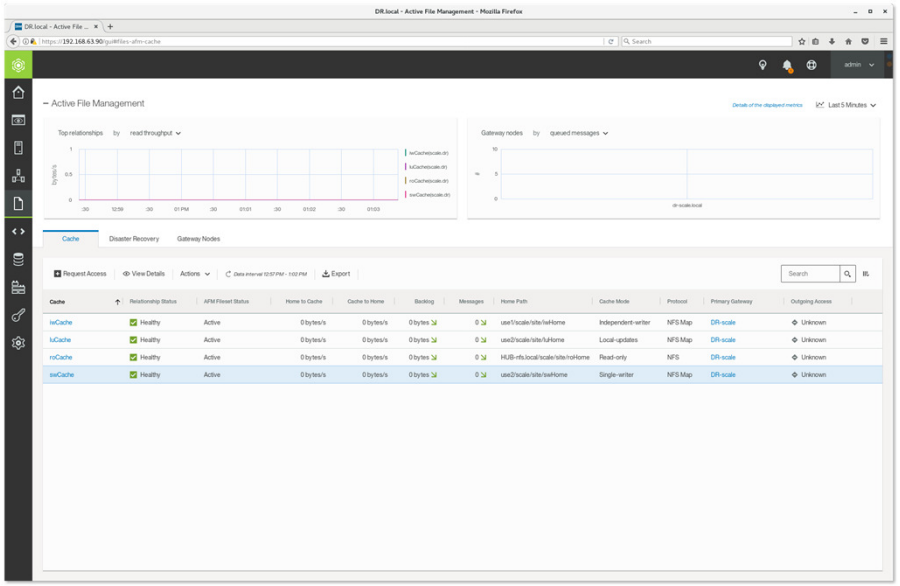
When the home is detected as available, the fileset is made active and the `afmHomeConnected` callback is triggered.

When a RO cache is disconnected, the `afmExpirationTimeout` setting indicates how long data is available. After that, cached data is considered “expired” and not available.

AFM and the Spectrum Scale GUI

The Spectrum Scale has several panes for monitoring AFM.

These require the performance monitoring framework to have AFM sensors activated.



Fine print: Interactions with Spectrum Scale features

Hierarchical Storage Management (HSM)

- This is supported in both cache and home, with both Spectrum Protect and Spectrum Archive.

Transparent Cloud Tiering

- This is not supported in the AFM cache.
- This is supported for an AFM home, but understand that reading or updating the file in cache may trigger a recall at the home.
  - There is no way to migrate directly between cache and cloud
  - Beware of tools reading file “thumbnails” on the cache triggering full file recalls on the AFM home, since AFM will transfer at least the first file system block.

Non-AFM filesets

- You can not link a fileset into a cache.
- A fileset under the home will appear to be a simple directory in the cache

Spectrum Scale for Windows

- AFM is not supported in clusters where some nodes run Spectrum Scale for Windows (not to be confused with protocol nodes exporting to Windows).

Not tested

- Quality of Service (QoS) with AFM cache, AFM-DR filesets
- IPv6 on gateway nodes

Not supported

- Gateway function on non-Linux nodes
- AFM through 3rd-party NFS servers (best-effort to support AFM, but not if the NFS server is suspect).

Other considerations

- Conflicts between SMB oplocks and AFM may delay updates of files exported through SMB.

Be sure to check the Spectrum Scale FAQ for current restrictions.

Working with AFM

When the fingers hit the keyboard …

Configuring gateway nodes

Use `mmchnode --gateway` to configure AFM gateway nodes.

Set recommended tuning parameters for gateway nodes with `mmchconfig` (make sure to limit scope to just the gateway nodes):

- pagePool=2G
- afmHardMemThreshold=40G
- afmDIO=2
- maxFilesToCache=10K
- afmMaxParallelRecoveries=3

Create any desired maps.

Enable AFM performance monitoring.

```
mmchnode --gateway -N REM-scale01,REM-scale02

mmchconfig pagePool=2G -N REM-scale01,REM-scale02
mmchconfig afmHardMemThreshold=40G \
    -N REM-scale01,REM-scale02
mmchconfig afmNumFlushThreads=8 -N REM-scale01,REM-scale02
mmchconfig maxFilesToCache=10K -N REM-scale01,REM-scale02
mmchconfig afmMaxParallelRecoveries=3
```

```
mmafmconfig add nfsmap1 --export-mapping \
    HUB-nfs1/REM-scale01,HUB-nfs2/REM-scale02

mmperfmon config update GPFSAFM.period=1
mmperfmon config update GPFSAFMFS.period=1
mmperfmon config update GPFSAFMFSET.period=1
```

Configuring the home file system, with NFS transport.

Ensure the file system path of the AFM home data is exported via NFS to the cache cluster's gateway nodes:

- Cluster Export Services
- cNFS
- Any NFS v3 server (can work with storage not in Spectrum Scale).

Be sure there is no "root squash"! Gateway nodes need to access the home as root.

If the data is stored in a Spectrum Scale file system, use `mmafmconfig` to enable passing ACLs, extended attributes, and sparseness to the AFM cache.

```
mmcrfileset scale.hub roHome
mmlinkfileset scale.hub roHome -J /scale/site/roHome
mmnfs export add /scale/site/roHome --client \
    '*(Access_Type=RW,Squash=no_root_squash,Protocols=v3)'
mmafmconfig enable /scale/site/roHome

mmcrfileset scale.hub iwHome
mmlinkfileset scale.hub iwHome -J /scale/site/iwHome
mmnfs export add /scale/site/iwHome --client \
    '*(Access_Type=RW,Squash=no_root_squash,Protocols=v3)'
mmafmconfig enable /scale/site/iwHome
```

Configuring AFM cache filesets

- We create cache filesets, associating them with AFM home targets, then link the filesets into the file system.
  - When using the `gpfs` (NSD) transport, the remote file system must be remote-mounted to the local cluster.
- At this point, the caches should be operational. Data is cached on demand.

```
mmcrfileset scale.rem roCache --inode-space new \
    -p afmTarget=nfsmap1:/scale/site/roHome \
    -p afmMode=read-only \
    -p afmNumFlushThreads=8 -p afmDIO=2
mmlinkfileset scale.rem roCache -J /scale/site/roCache

mmcrfileset scale.flash iwCache --inode-space new \
    -p afmTarget=gpfs:///scale/site/iwHome \
    -p afmMode=independent-writer \
    -p afmNumFlushThreads=8 -p afmDIO=2
mmlinkfileset scale.flash iwCache -J /scale/flash/iwCache
```

Prefetching data

Both metadata and data may be prefetched into the cache.

On the home side, prepare a list of files that should be prefetched, created using:

- `find -noleaf`
- `ls -lR`
- The policy engine (a “list” policy)

```
RULE 'list-home'
    LIST 'outlist'
    DIRECTORIES_PLUS
```

On the cache, prefetch with `mmafmctl`.

On the home cluster:

```
mmapplypolicy scale.hub \
    -P /root/list-pol.txt  -I defer -f
/tmp
scp /tmp/list.outlist DR-scale-wan:/tmp/
```

On the cache cluster:

```
mafmctl scale.dr prefetch -j iwCache \
    --list-file /tmp/list.outlist \
    --home-fs-path /scale/site/iwHome
```

Working with AFM-DR

Oops…

AFM-based Active/Passive Asynchronous DR

AFM-DR extends the AFM Single Write cache model to have a writeable primary fileset associated with a read-only secondary fileset.

Data can be trucked to preload the secondary fileset.

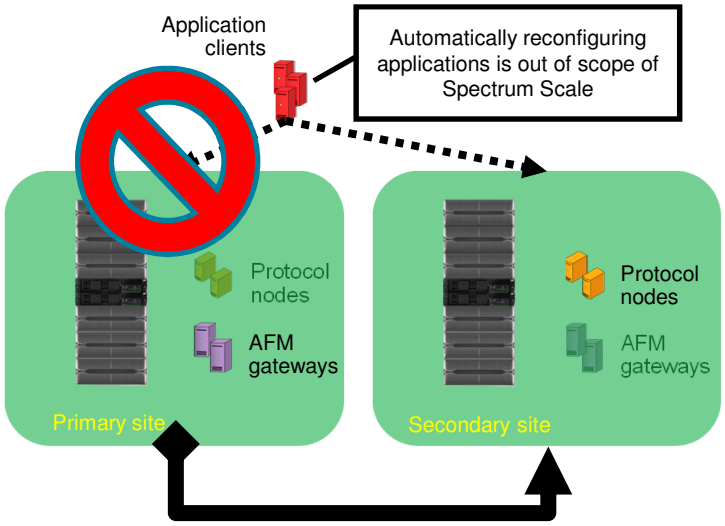
Data written to the primary fileset is constantly being replicated to the secondary fileset, asynchronously.

- *Optionally*, a regular "peer snapshot" can be taken at a configurable interval, based on an RPO.
- When promoting the secondary to be the acting primary, it can be rolled back to this snapshot.
- This is mostly useful when there is some risk of out-of-order updates leading to a corrupted acting primary.

Upon failover, the secondary site becomes the acting primary site (and the fileset becomes writeable).

Upon failback, the original primary is updated with all the changes made to the acting primary (original secondary). Then the secondary is converted back to a true secondary.

If necessary, a new primary or secondary site can be established.



Configure AFM-DR

1. At the primary site, create NFS maps and the primary fileset.
   - For the `afmTarget`, use the location the secondary will have once it is created.
   - Capture the ”PrimaryID”, which is needed in the next step.
   - Do not link the primary fileset yet!

2. At the *secondary* site, create the secondary fileset, link it, and export it through NFS.
   - Set `afmPrimaryID` to the value you captured in the last step.

3. At the *primary* site, link the primary fileset.

On the primary:

```
mmafmconfig add drmap --export-map 'DR-nfs1/HUB-scale1,DR-
nfs2/HUB-scale2'
mmcrfileset scale.hub primary --inode-space new \
    -p afmMode=primary \
    -p afmTarget=drmap:/scale/site/secondary \
    -p afmNumFlushThreads=8 -p afmDIO=2
PRIMARYID=$(mmafmctl scale.hub getPrimaryID -j \
               primary | awk '{print $NF}’)
```

On the secondary: (transfer value of `PRIMARYID`)

```
mmcrfileset scale.dr secondary --inode-space new \
    -p afmMode=secondary \
    -p afmPrimaryID=${PRIMARYID} \
    -p afmNumFlushThreads=8
mmlinkfileset scale.dr secondary \
    -J /scale/site/secondary
mmnfs export add /scale/site/secondary --client \
    '*(Access_Type=RW,Squash=no_root_squash,Protocols=v3)'
mmafmconfig enable /scale/site/secondary
```

Back on the primary:

```
mmlinkfileset scale.hub primary \
    -J /scale/site/primary
```

Configure AFM-DR (trucking data)

- Start with independent filesets at both sites, containing the same data.
  - Primary fileset will be `truck`
  - Secondary fileset will be `dock`

1. Get the “PrimaryID” of the to-be primary fileset.

2. Convert `dock` to be a secondary fileset, and export it through NFS.
   - Associate the fileset with the “PrimaryID” of the to-be primary fileset.

3. Convert `truck` to be a primary fileset
   - Set its `afmTarget` to be the location of `dock`, through NFS.

On the primary site:

```
PRIMARYID=$(mmafmctl scale.hub getPrimaryID \
    -j truck | awk '{print $NF}’)
```

On the secondary: (transfer value of `PRIMARYID`)

```
mmafmctl scale.dr convertToSecondary \
    -j dock --primaryid PRIMARYID
mmnfs export add /scale/site/dock --client \
   '*(Access_Type=RW,Squash=no_root_squash,Protocols=v3)’
```

On the primary site:

```
mmafmctl scale.hub convertToPrimary -j truck \
    --afmTarget=drmap:/scale/site/dock --inband
mmafmctl scale.hub getState
```

On a DR event

The primary site is down!

1. Convert the secondary filesets to be primary filesets. (Optionally restore to last RPO snapshot.)
2. Point applications to the secondary fileset.

The secondary site is down!

- Once the secondary site is back up, the primary will keep on syncing.

The secondary site is permanently down!

- It is possible to create a new secondary site.

On the secondary site:

```
mafmctl scale.dr failoverToSecondary \
    -j secondary
mmafmctl scale.dr failoverToSecondary \
    -j dock
```

Failback from a primary site failure

1. Put primary filesets into failback mode.

2. Apply updates.
   - The secondary fileset is still an acting primary, and applications may continue updating it.
   - Repeat applying updates, until there is not much to update.

3. Stop applications! Yes, this is an outage.
   - Stop the failback operation now.

4. Unlink secondary filesets, convert them back to secondaries, and relink them.

Working on the primary site:

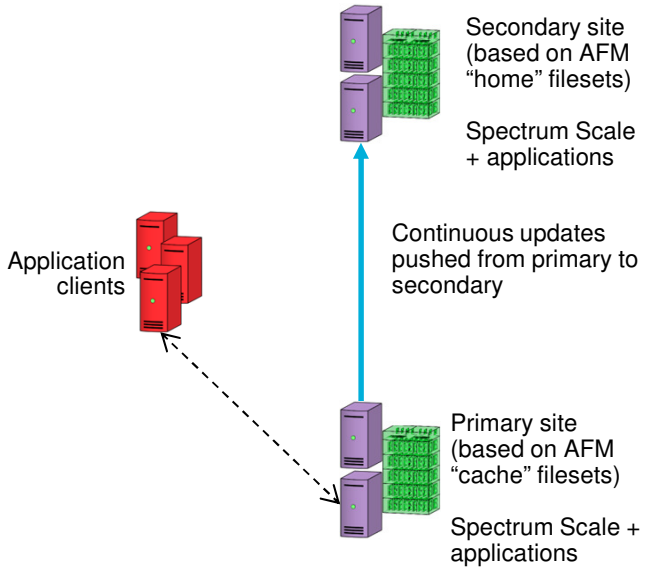
```
mafmctl scale.hub failbackToPrimary \
    -j primary --start
mmafmctl scale.hub failbackToPrimary \
    -j truck --start
mmafmctl scale.hub applyUpdates -j primary
mmafmctl scale.hub applyUpdates -j truck
# After applications are stopped…
mmafmctl scale.hub failbackToPrimary \
    -j primary --stop
mmafmctl scale.hub failbackToPrimary \
    -j truck --stop
```

Working on the secondary site: (after turning off applications)

```
mmunlinkfileset scale.dr secondary -f
mmunlinkfileset scale.dr dock -f
mmchfileset scale.dr secondary -p afmMode=secondary \
    -p afmPrimaryID=${PRIMARYID_primary}
mmchfileset scale.dr dock -p afmMode=secondary \
    -p afmPrimaryID=${PRIMARYID_truck}
mmlinkfileset scale.dr secondary \
    -J  /scale/site/secondary
mmlinkfileset scale.dr dock \
    -J /scale/site/dock
```

Some thoughts on implementing AFM-DR

- Current limitations are changing rapidly – be sure to check with Development.
  - Currently there is a review process needed to get a key needed to activate AFM-DR.
- Use the latest version of Spectrum Scale to get the best performance and reliability.
- This is active/passive – if you use case is active/active, you must choose another replication solution!
- Protocol nodes can be set to be easily failed over to the secondary site too



Engineering guidance for AFM-DR (applicable to AFM)

Filesets:

- Maximum 100 filesets
- Maximum 100 million files / fileset
- Be aware that AFM filesets see a performance degradation of 1.2x to 4x, depending on gateway node performance and the networking between them and Scale clients.
- When converting independent filesets, use in-band trucking.

Gateways:

- Rough maximum 10 filesets / gateway node
- Gateway node should be dedicated to this function. **It must not be a protocol node.**
- 128GB RAM
- The /var file system needs storage for (number of filesets)*(files/fileset)*255 bytes.
- Adequate CPU: the more filesets, the more horsepower needed.

Network:

- Well-tuned AFM-DR can use 80% of available intersite bandwidth.
- Network between Scale clients and local gateways is critical.

Tunables:

- If you do not need the RPO snapshot (usually you don't), then disable it.
- Other settings:
  - `pagePool=2G`
  - `afmHardMemThreshold=40G`
  - `afmNumFlushThreads=8`
  - `afmDIO=2`
  - `maxFilesToCache=10K`
  - `afmMaxParallelRecoveries=3`

Further resources

"Of making many books there is no end…" – Ecc. 12:12

Further resources

- IBM Knowledge Center – https://www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html
- Spectrum Scale FAQ – https://www.ibm.com/support/knowledgecenter/en/STXKQY/gpfsclustersfaq.html
- developerWorks Spectrum Scale Wiki (AFM) – https://www.ibm.com/developerworks/community/wikis/home?lang=en-us - !/wiki/General Parallel File System (GPFS)/page/Active File Management (AFM)

Accelerate with IBM Storage Webinars

The Free IBM Storage Technical Webinar Series Continues in 2018...

Washington Systems Center – Storage experts cover a variety of technical topics.

Audience: Clients who have or are considering acquiring IBM Storage solutions. Business Partners and IBMers are also welcome.

To automatically receive announcements of upcoming Accelerate with IBM Storage webinars, Clients, Business Partners and IBMers are welcome to send an email request to accelerate-join@hursley.ibm.com.

Located in the Accelerate with IBM Storage Blog:
https://www.ibm.com/developerworks/mydeveloperworks/blogs/accelerate/?lang=en

Also, check out the WSC YouTube Channel here:
https://www.youtube.com/playlist?list=PLSdmGMn4Aud-gKUBCR8K0kscCiF6E6ZYD&disable_polymer=true

2018 Webinars:

January 9 – DS8880 Easy Tier
January 17 – Start 2018 Fast! What's New for Spectrum Scale V5 and ESS
February 8 - VersaStack - Solutions For Fast Deployments
February 16 - TS7700 R4.1 Phase 2 GUI with Live Demo
February 22 - DS8880 Transparent Cloud Tiering Live Demo
March 7 - Spectrum Storage Management, Control, Insights, Foundation; what's the difference?
March 15 - IBM FlashSystem A9000/R and SVC Configuration Best Practices
March 27 - IBM FlashSystem A9000/R Technical Update
April 12 - Introducing Spectrum NAS - The Newest Member in the Spectrum Storage Family
April 26 - TS7700 Grid Configuration Changes -- Joins, Merges and Removals
May 8 - DS8880 Technical Update
June 7 – Economic Value of Response Time
June 21 – Deep Dive into Spectrum Scale AFM
June 28 - Open System Tape, What's New

Register Here:
https://ibm2.webex.com/ibm2/onstage/g.php?MTID=e74c4d48c903c2dbff250554ee538c0e4

July 19 - Copy Services Manager Version 6.2.2 Technical Update

Register Here:
https://ibm2.webex.com/ibm2/onstage/g.php?MTID=ea05fe90ed7965172d59f733262123b43

August 2 - Back to Basics - TS7700 Concepts and Operations

Register Here:
https://ibm2.webex.com/ibm2/onstage/g.php?MTID=ed3ae82187b2aaad0c7dc1bdfa459aed4