Archive Fabric Module Whitepaper

Table of Contents

Introduction

The exponential growth of unstructured data collections continue to present both cost and management challenges for organizations generating and managing large amounts of data. The first step toward controlling storage costs is to establish a scalable, cost efficient archive. Once the basic archive infrastructure is in place, the next challenge is how to identify and move data from a variety of different storage silos into the archive.

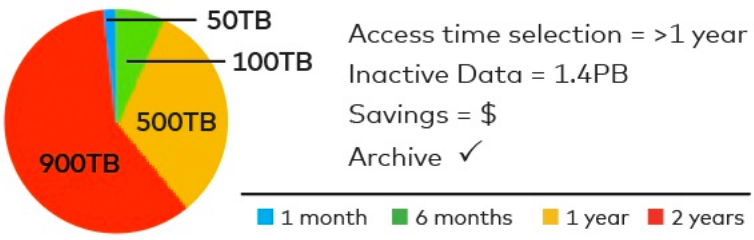
Many filesystem and archive management solutions, including Versity Storage Manager (VSM) provide integrated data analysis and movement features within their own software stack. However, very few products are designed to interoperate with external systems. Most customer environments host a number of different storage products in order to avoid the dangers of becoming overly reliant on one technology or vendor and also to take advantage of different features and capabilities.

In order to leverage a central archive resource, it is often necessary to deploy tools for moving data from a range of different storage systems.

This white paper provides a technical overview of the Versity solution for analyzing and moving data from external filesystems.

Overview

The **Versity Archive Fabric Module™** (AFM) utility analyzes external filesystems, identifying infrequently used data sets that are strong candidates for archiving. A browser based report presents statistics based on file age, access time, modification time, and other file attributes. This information helps the users or administrators identify data that should be moved to the archive tier. AFM also orchestrates the movement of data from primary storage systems to archival storage systems.



AFM is a powerful and easy to use tool that can help free space on primary storage. It does this with two easy to use functions: identification and movement.

AFM Operation

AFM moves data from external filesystems to VSM in one of three operational modes:

Invisible Mode

Moves data to the archive transparently, leaving behind the original namespace by using symbolic links.

Mover Mode

Moves data to the archive, performs integrity validation, confirms archive copies then deletes source data.

Replication Mode

Copies all data to a second location for disaster recovery - namespace and data replication.

Scanning Target Filesystems

Scan
Required flags: -scan -fs <target path>

The AFM utility will scan the target filesystem and make statistics available by serving a webpage on http://localhost:8080 that can be accessed using a local web browser. Currently this functionality requires an active internet connection on the host running the web browser.

Replicating Target to Archive

Migrate
Required flags: -migrate -fs <target path> -archive <archive path>

The AFM utility will replicate all files from the target file system to the archive. This has the advantage of being able to recover the entire target file system in case of disaster. Optionally, the user can define min/max thresholds for file attributes (atime, mtime, and size) that will tell AFM to replace the original file in the target file system with a symlink to the archive file. By default all files are replicated but no files are replaced with symlinks.

The AFM utility is intended to be run periodically to keep the target file system and archive in sync with each other. To prevent unnecessary data movement, and file will only be copied if:

- not already in the archive
- if in the archive and has either different mtime or size than the target file

This behavior can be changed by using the -force option to force copying all files to the archive even if file mtime and size are the same.

Log File

A log file is created in each directory to track which files within that directory have been replicated along with checksum. The log entries have the format:
hash_type, hash_string, mtime, migrate_time, filename
The log filename by default is ".afmlog", but can be changed with the -logfile option.

Checksums

When migrating files from target filesystem to archive, a checksum is calculated and logged in the log file. The default checksum is sha256, but md5, sha1, sha384, sha512 are also supported. This can be changed with the -hash option.

Replacing Target Files with Symbolic Links

While replicating the target namespace to the archive, the AFM utility can replace files in the target namespace matching specified criteria with symlinks to the corresponding files in the archive. This has the benefit of freeing up space in the target filesystem while maintaining file availability by redirecting access to the archive file.

The criteria for replacing files with symlinks is based on the following attributes:
- atime max with the -amax option (will match file less than this value)
- atime min with the -amin option (will match file greater than this value)
- mtime max with the -mmax option (will match file less than this value)
- mtime min with the -mmin option (will match file greater than this value)
- size max with the -smax option (will match file less than this value)
- size min with the -smin option (will match file greater than this value)

The times for atime/mtime min/max are specified with relative times using the format Y,M,D. For example, -amin 1,2,3 will match all files with access time older than 1 year 2 months and 3 days ago.

When specifying multiple criteria options, and file must match all options for it to be replaced with a symlink.

There is a special skipfile that when present will prevent anything within that directory from being replaced with a symlink. This can allow uses to define important or active directories that should always remain in the target filesystem. By default the file is ".nomigrate", but can be changed with the -skipfile option.

Important notes for NFS/SMB clients:

For an NFS client to correctly resolve symlinks, both the target file system and archive filesystem must be mounted. The AFM utility can use either relative or direct symlinks. If using relative links, the client must mount the filesystem at similar relative paths to host where AFM is executed. If using absolute link paths, the client must mount both file systems at the same mount point as the host executing AFM.

The SMB server can be configured to follow symbolic links in which case the client would only need to mount the target file system. Otherwise the same considerations as specified above for NFS apply as to SMB as well.

Rehydrating Target from Archive

Hydrate

Required flags: -hydrate -fs <target path> -archive <archive path>

The AFM utility can rehydrate the target file system from the archive namespace. The amin, amax, mmin, mmax, smin, and smax options can be used when rehydrating the target filesystem to prevent unnecessary data copies. For files matching the criteria, a symlink will be created to the archive file instead of replicating the file contents back to the target.

AFM Examples

In the following examples, the “primary” file system represents the target file system and has been pre-populate with some test files.

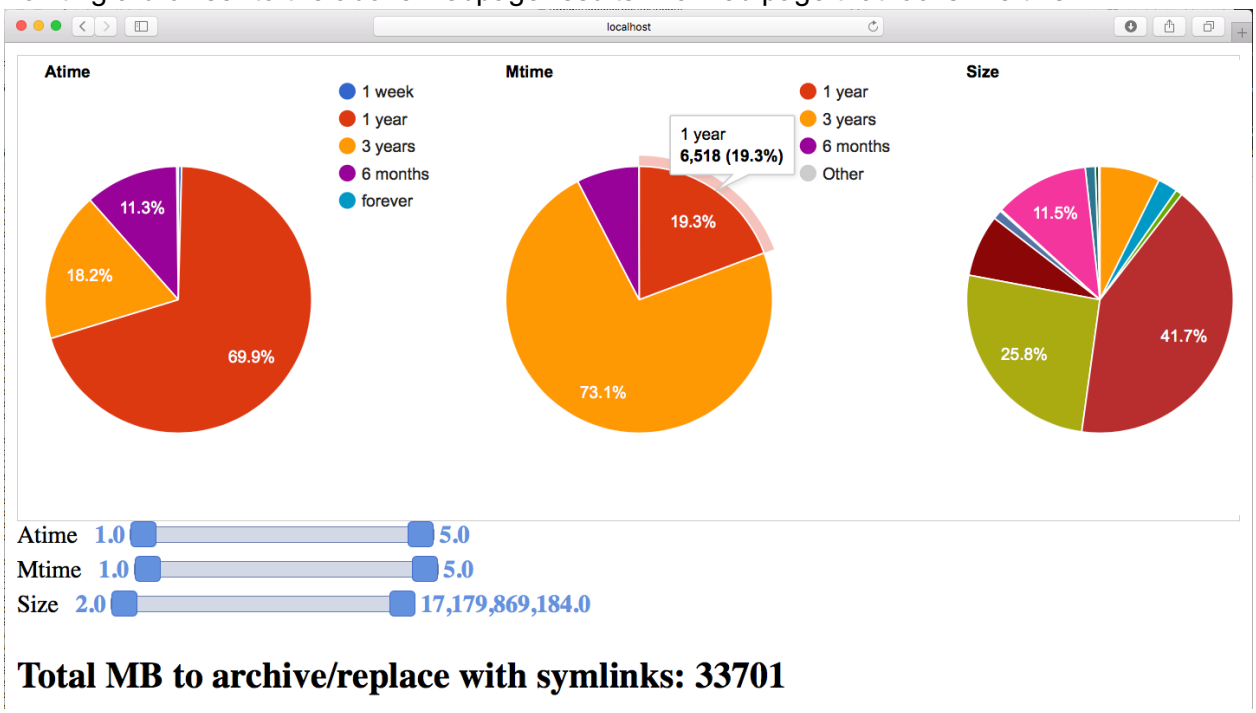
Example Scan

This example creates a skipfile in one of the directories to show “Non-Migratable” size in scan output.

```
$ touch ~/primary/vsm-1.4.3/.nomigrate
$ afm -scan -fs ~/primary
Total Files: 149
Total Directories: 16
Total SymLinks: 0
Total Non Regular Files: 0
Total Non-Migratable Size: 6MB

point browser to http://localhost:8080
ctrl-c to quit
```

Pointing a browser to the above webpage results in a web page that looks like this:



Example Migrate

This example replicates all files from target file system to archive file system, and replaces files larger than 1MB in the target file system with a symlink to the archive file.

```
$ afm -smin 1048576 -fs ~/primary/ -migrate -archive ~/archive/
```

Listing one of the directories with both large and small files shows the files that were replaced with symlinks.

```
$ ls -lah ~/primary/vsm-1.4.2
total 680
drwxr-xr-x  16 ben  staff   544B Jan 24 10:14 .
drwxr-xr-x  15 ben  staff   510B Jan 24 10:13 ..
-rw-r--r--   1 ben  staff   8.1K Jan 24 10:13 Changelog.txt
-rw-r--r--   1 ben  staff   2.8K Jan 24 10:13 README.vfs-vsm
-rw-r--r--   1 ben  staff   1.7K Jan 24 10:13 RPM-GPG-KEY-vsm
-rw-r--r--   1 ben  staff    40K Jan 24 10:13 ReleaseNotes.txt
-rw-r--r--   1 ben  staff   561B Jan 24 10:13 hash.md5
-rw-r--r--   1 ben  staff    61K Jan 24 10:13 libs3-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    62K Jan 24 10:13 libs3-devel-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    31K Jan 24 10:13 libvsm-1.0.0-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    39K Jan 24 10:13 libvsm-debuginfo-1.0.0-1.el6.x86_64.rpm
lrwxr-xr-x   1 ben  staff    55B Jan 24 10:14 vsm-1.4.2-1.el6.x86_64.rpm ->
/Users/ben/archive/vsm-1.4.2/vsm-1.4.2-1.el6.x86_64.rpm
```

```
lrwxr-xr-x   1 ben  staff    65B Jan 24 10:14 vsm-debuginfo-1.4.2-1.el6.x86_64.rpm ->
/Users/ben/archive/vsm-1.4.2/vsm-debuginfo-1.4.2-1.el6.x86_64.rpm
```

Listing the directory where the skipfile was created shows how no files in that directory were replaced with symlinks.

```
$ ls -lah ~/primary/vsm-1.4.3
total 14032
drwxr-xr-x  17 ben  staff   578B Jan 24 10:14 .
drwxr-xr-x  15 ben  staff   510B Jan 24 10:13 ..
-rw-r--r--   1 ben  staff     0B Jan 24 10:14 .nomigrate
-rw-r--r--   1 ben  staff   8.6K Jan 24 10:13 Changelog.txt
-rw-r--r--   1 ben  staff   2.8K Jan 24 10:13 README.vfs-vsm
-rw-r--r--   1 ben  staff   1.7K Jan 24 10:13 RPM-GPG-KEY-vsm
-rw-r--r--   1 ben  staff    42K Jan 24 10:13 ReleaseNotes.txt
-rw-r--r--   1 ben  staff   561B Jan 24 10:13 hash.md5
-rw-r--r--   1 ben  staff    61K Jan 24 10:13 libs3-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    62K Jan 24 10:13 libs3-devel-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    31K Jan 24 10:13 libvsm-1.0.0-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff    39K Jan 24 10:13 libvsm-debuginfo-1.0.0-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff   4.3M Jan 24 10:13 vsm-1.4.3-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff   2.2M Jan 24 10:13 vsm-debuginfo-1.4.3-1.el6.x86_64.rpm
```

Example Hydrate

This example rehydrates the primary file system from the archive. Using the same smin option as the migrate, we can maintain the behavior of files over 1MB being rehydrated as symlinks to prevent unwanted data movement.

```
$ rm -rf ~/primary/*
$ ls -la ~/primary/
total 0
drwxr-xr-x   2 ben  staff    68 Jan 24 10:16 .
drwxr-xr-x+ 97 ben  staff  3298 Jan 24 10:13 ..
$ afm -smin 1048576 -fs ~/primary/ -hydrate -archive ~/archive/
```

Listing a directory shows the combinations of file and symlinks that were rehydrated from the archive.

```
$ ls -la ~/primary/vsm-1.4.2
total 584
drwxr-xr-x  15 ben  staff    510 Jan 24 10:17 .
drwxr-xr-x  15 ben  staff    510 Jan 24 10:17 ..
-rw-r--r--   1 ben  staff   8277 Jan 24 10:13 Changelog.txt
-rw-r--r--   1 ben  staff   2819 Jan 24 10:13 README.vfs-vsm
-rw-r--r--   1 ben  staff   1744 Jan 24 10:13 RPM-GPG-KEY-vsm
-rw-r--r--   1 ben  staff  41363 Jan 24 10:13 ReleaseNotes.txt
-rw-r--r--   1 ben  staff    561 Jan 24 10:13 hash.md5
-rw-r--r--   1 ben  staff  62460 Jan 24 10:13 libs3-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff  63116 Jan 24 10:13 libs3-devel-3.1.1-1.el6.x86_64.rpm
-rw-r--r--   1 ben  staff  31356 Jan 24 10:13 libvsm-1.0.0-1.el6.x86_64.rpm
```

```
-rw-r--r--   1 ben  staff  39756 Jan 24 10:13 libvsm-debuginfo-1.0.0-1.el6.x86_64.rpm
lrwxr-xr-x   1 ben  staff     55 Jan 24 10:17 vsm-1.4.2-1.el6.x86_64.rpm ->
/Users/ben/archive/vsm-1.4.2/vsm-1.4.2-1.el6.x86_64.rpm
lrwxr-xr-x   1 ben  staff     65 Jan 24 10:17 vsm-debuginfo-1.4.2-1.el6.x86_64.rpm ->
/Users/ben/archive/vsm-1.4.2/vsm-debuginfo-1.4.2-1.el6.x86_64.rpm
```

Example AFM Log

This is an example of what the AFM log entries look like. Lines wrapped in documentation, but are on a single line in the log file.

```
$ head -1 ~/primary/vsm-1.4.2/.afmlog
sha256 b14465f5897e431f67cc88f106179b21a43e570000c2cdd21c7d96cbd57a3078 2018-01-24
10:20:46 -0800 PST 2018-01-24 10:20:49.281150281 -0800 PST m=+0.062560436 README.vfs-
vsm
```

The entry is: hash_type, hash_string, mtime, migrate_time, filename
The hash type is for future verification to indicate hash type of the has string.
The hash string is a hexadecimal encoding of the hash value for the file contents.
For now the time formats are: "2006-01-02 15:04:05.999999999 -0700 MST" and a final field "m=±<value>" for migrate time, where value is the monotonic clock reading formatted as a decimal number of seconds.

Conclusion

Versity's Archive Fabric Module™ (AFM) is an easy to use tool that helps relieve pressure on primary filesystems by identifying and moving files that are good candidates for archiving. It is free for Versity customers to use as part of their VSM subscription.

Contact us at sales@versity.com or https://www.versity.com/contact for more information or a demo.

About Versity

Versity is an independent, software-defined mass storage company focused on rapid innovation and long-term growth. We build scalable, modular and efficient exabyte-scale data storage solutions and aggressively invest in new technology. Our customer-friendly business model, exclusive focus on mass storage and highly rated customer support set us apart from the legacy storage vendors.