AEM Storage Considerations

Ian Reasor, Technical Architect, Adobe Partner Experience

Introduction

Choosing the right storage architecture for your AEM deployment can ensure that your system is performant and reliable. Many factors go into choosing the appropriate storage for your deployment, and there are many moving parts to account for. We will look at choosing the right storage for your SegmentNodeStore in TarMK deployments, your FileDataStore, and your temp/cache directories. Finally, we will look at use cases in which using a shared FileDataStore or S3 datastore is appropriate, and the hardware considerations for these deployments.

SegmentNodeStore

AEM's SegmentNodeStore is vital to the performance of any TarMK instance as it is the persistence layer for AEM's database. For this reason, we recommend that the SegmentNodeStore is always stored on local solid state disks or a SAN. Some have asked about storing their SegmentNodeStore on a NAS (using NFS or SMB). We do not recommend this approach due to increased latency and the inability of the operating system to memory-map files in this type of deployment, which TarMK relies on heavily. Storing a SegmentNodeStore on a SAN with a local filesystem, however, is acceptable.

FileDataStore

We recommend configuring all production AEM deployments to use either a FileDataStore or an S3 datastore. The datastore is responsible for storing all binaries that exceed the defined size threshold, which defaults to 4KB. Unlike the SegmentNodeStore, it is appropriate to leverage a NAS (NFS or SMB) to host your FileDataStore, mounted by the operating system as a local drive. When using external storage to host your datastore, ensure that you are using high-performance devices and that you have sufficient network bandwidth between the AEM instance and the storage device to avoid any performance issues.

Shared Datastores

Some customers opt to share their datastore between multiple instances. There are two deployment scenarios where we recommend this as an option: when using multiple author instances and when publishing large amounts of binaries—as is often the case in AEM Assets implementations that are leveraging the Asset Share feature.

There are three possible deployments of AEM that will leverage multiple author instances. When using MongoMK, we recommend configuring a shared datastore between the instances to ensure optimal performance of the Mongo database. When using TarMK cold standby, some customers opt to share a datastore between the author environments to lower storage costs, while others choose to have each environment maintain a separate copy of the binaries to improve overall redundancy. Finally, when

using workflow offloading, we recommend using a shared datastore in concert with binaryless replication between the author and worker instances.

When replicating large amounts of binary data, some customers opt to share a datastore between their author and publish instances. When used along with binaryless replication, this allows the AEM author environment to send only the asset metadata in publish operations; this greatly reduces the time and resources required to activate assets and lowers the storage footprint of these assets.

Shared FileDataStores

A shared FileDataStore is simply a FileDataStore that is stored on a networked device and mounted on multiple instances. Outside of datastore garbage collection, AEM does not interact with a shared FileDataStore any differently than it would if it were the sole instance connected to the datastore. The implication of this is that concurrent access to files on this shared volume must be managed by an operating system or filesystem. While shared disk filesystems like GFS and GPFS can manage this for a SAN deployment, in practice, we seldom see these in use. As a result, most customers will opt for a NAS/NFS approach.

S3 Datastores

S3 datastores are managed in the Amazon cloud and thus do not require the level of storage architecture consideration that a FileDataStore requires. However, there are pros and cons for on-premise customers to consider when deciding to a datastore in S3. The benefits to S3 are a lower cost, the ability to expand your storage at will, and having the infrastructure managed by a third-party provider. The downside to S3 is higher latency when connecting to the datastore. All customers of Adobe Managed Services who have a large datastore will have an S3 datastore provisioned by default.

When using binaryless replication, such as in offloading scenarios, there are potential issues with S3. Due to the nature of the local write cache leveraged by the S3 datastore, it is possible for a binaryless replication message to be received by the offloading instance before the binary is fully written to the S3 bucket. In this case, the offloading instance responds to the primary instance with a message indicating that the replication should be redone with the binary inline. While this allows for processing on the offloading node to proceed, the resubmitted replication message carries additional overhead and is less efficient than using offloading with a FileDataStore.

Temp and Cache Directories

The final point of consideration for storage is in the placement of our cache and temp directories. AEM will store cache directories under the crx-quickstart folder when they are configured. We do recommend configuring caches for the FileDataStore when in use and Oak indexes when using Mongo MK. Because the purpose of these caches is to improve performance, it is important that they are stored on high-performance disks, and we recommend solid state drives for this purpose. AEM's temp directory location can be configured through JVM switches in the startup command, and we recommend pointing this directory to an ephemeral drive. Because this drive is stored in memory rather than on disk, it will make operations that use this directory, such as asset upload and manipulation, much more performant.

Backup and Maintenance Considerations

AEM's datastores implement an append-only model of data persistence. While this allows for faster writes to the datastore, it also means that when files are modified or deleted that the old version is not removed from disk. To prevent a datastore from continually growing, AEM provides a datastore garbage collection feature to prune old unreferenced blobs from the datastore.

When planning for garbage collection, it is important to understand the relationship between the node store and the data store. Nodes in the node store reference the blobs in the data store. If a node references a blob that is missing, the system will be unstable and potentially unusable. This is normally not a problem but can become one if an appropriate backup strategy has not been implemented. Should garbage collection be executed and then a backup of the node store from before garbage collection was executed needs to be restored, it is likely that there will be nodes in the node store pointing to missing blob files.

To better illustrate this scenario, consider the following example:

On March 1, a backup is run of our system's node store.
On March 2, a user deletes an unneeded asset from the repository.
On March 3, compaction and garbage collection are run on our instances, deleting the asset's binary from the datastore.
On March 4, we need to restore the backup of the node store taken on March 1. This backup contains a reference to the binary that was removed from the datastore on March 3.

For this reason, it is important that you plan to back up your datastore before running any garbage collection operations. In the case of large repositories, this can have a substantial impact on the amount of storage that will be required as it essentially doubles the size of storage required for the datastore. While we recommend high-performing disks for the datastore itself, the backup can be stored on lower-performing hardware or in a lower-cost cloud-based solution such as Amazon Glacier.

Storage Matrix

The following table illustrates the appropriate storage options by store type:

Type	RAM Disk	Local Disk (SSD)	Local Disk (Magnetic)	SAN	NAS
SegmentNodeStore		X		X	
FileDataStore (single)		X	X	X	X
FileDataStore (shared)				*	X
Temp and Cache Directories	X				

**shared filesystem only*

Additional Resources

https://cqdump.wordpress.com/2016/02/23/tarmk-on-nas/

https://cqdump.wordpress.com/2016/02/24/tarmk-and-san/

https://helpx.adobe.com/experience-manager/6-4/sites/deploying/using/data-store-config.html

https://helpx.adobe.com/experience-manager/6-4/assets/using/assets-sizing-guide.html

https://helpx.adobe.com/experience-manager/6-4/assets/using/performance-tuning-guidelines.html

https://helpx.adobe.com/experience-manager/6-4/assets/using/assets-offloading-best-practices.html