ObjectMatrix

Media Data Replication Using MatrixStore Technologies

Object Matrix Ltd
Your trusted storage partner

www.object-matrix.com
info@object-matrix.com
+44(0)2920 382 308

Introduction

The Modern Day Digital Media Workflow

Within media workflows there are three tiers of storage *"traditionally"* spoken of:

- *"Online"* tier: Online, on premises storage. Used for editing and also at times for play-out. Typically this is a NAS or SAN device.
- *"Offline"* tier: Offline storage sometimes found on premises and sometimes off premises. Often this is a tape solution – LTO tapes or tapes that are actually direct from the cameras.
- *"Nearline"* tier: more modern media workflows have a Nearline layer – this is a place where data can be collected and easily managed within a scalable solution. Object Matrix's MatrixStore disk-based solution can be used for this (though of course other Nearline solutions from other vendors are available).

But, in terms of a neat and tidily arranged *"three tiers of storage"* the world has changed. The reality is far more that there are a myriad of different storage devices around an organisation and each one is tightly tuned to a set of tasks.
The driver for this is that the sheer amount and variety of digital assets created has vastly increased. There are now a plethora of geographical dispersed places where assets are created on all kinds of devices. These assets need to be ingested, edited and distributed: but no longer even necessarily in that order. Flexibility is key and an array of storage solutions reflect that but for the purpose of manageability data is generally driven into a large or set of large storage pools – whatever tier you want to call that.

However, simply putting the assets into a large storage pool has certain challenges:

- Dangers of mistaken or malicious deletes – deletes that can wipe a set of data in a few milliseconds – deletes that can be instigated by disgruntled employees, viruses or hackers alike
- Dangers from power failures or mass equipment malfunctions
- The requirement to continue production / transmission from another site if a single location was to be compromised

So at some point, sooner or later, every organisation starts to consider the benefits of keeping an offsite copies of data, for disaster recovery and/or business continuity reasons alone, assets are simply too valuable to be risked in a single location. And probably the most popular solution to achieve offsite copies of data comes under the banner of *"data replication"*.

Media Data Replication Using MatrixStore Technologies

This document discusses the various different requirements in Business Continuity (**BC**) and Disaster Recovery (**DR**) workflows, and, the replication models that can be applied via MatrixStore to achieve BC and DR. The aim of the document is to enable the reader to implement an appropriate strategy in their organisation for the short and long term safe keeping of data in geographically dispersed locations and possibly to additionally enable fast access and usage of assets across those locations.

What is Data Replication?

Data replication is the replication of data from one computer system to another. Normally the two systems are geographically separated and normally the systems are loosely coupled, i.e., one is not dependent on the other for its operations.

Whilst simple in concept implementations vary widely in their strengths and weaknesses. Data consistency (which is taken for granted on a local storage) is rarely maintained across geographically separated systems unless significant compromises are made.

Disaster Recovery vs Business Continuity vs Wide Area Nearline

These terms often get intermixed but they are not the same thing:

Disaster Recovery

With MatrixStore a Disaster Recovery Plan (**DRP**) refers to how to access assets in the case of a local data loss or loss of access to data. This could include keeping an offsite copy of data so that data can be recovered. A DRP could be just a backup of the essential data, e.g., the high-res. Also, a DRP may make compromises such as only keeping an essential sub-set of the complete data set. The DRP answers the question: *"If we lost all the **data** at site x how would we recover or continue?"*

Business Continuity

However, a Business Continuity Plan (**BCP**) should be formed if the question is *"If we lost the entire **building** at site x how would we recover or continue?"* This usually involves, in the production world, having editing facilities at the *"failover"* site as well as the assets from site.

Wide-Area Nearline

Because having a complete second site just sitting around doing nothing and waiting for a disaster is an expensive resource and because the reality in many organisations is that they have many hubs of operation that effectively become the *"BCP"* if one of the other hubs was to become inactive for any reason, then for many a BCP should be multi-directional: e.g., site x is the BCP for site y, but equally, site y is the BCP for site x. Furthermore, in this topology, hubs can start to become collection points for assets that are passed on to other hubs, 24/7 editing can be set-up, etc. For media workflows Object Matrix has coined this topology as *"Wide-Area-Nearline"* (**WAN**). The ability to ingest and work on assets at any location and at any point.

Business Continuity

Whether the plan is to have a simple backup of a piece of critical data in a DRP, a more complex BCP or an all-singing all-dancing WAN you are going to need to replicate data.

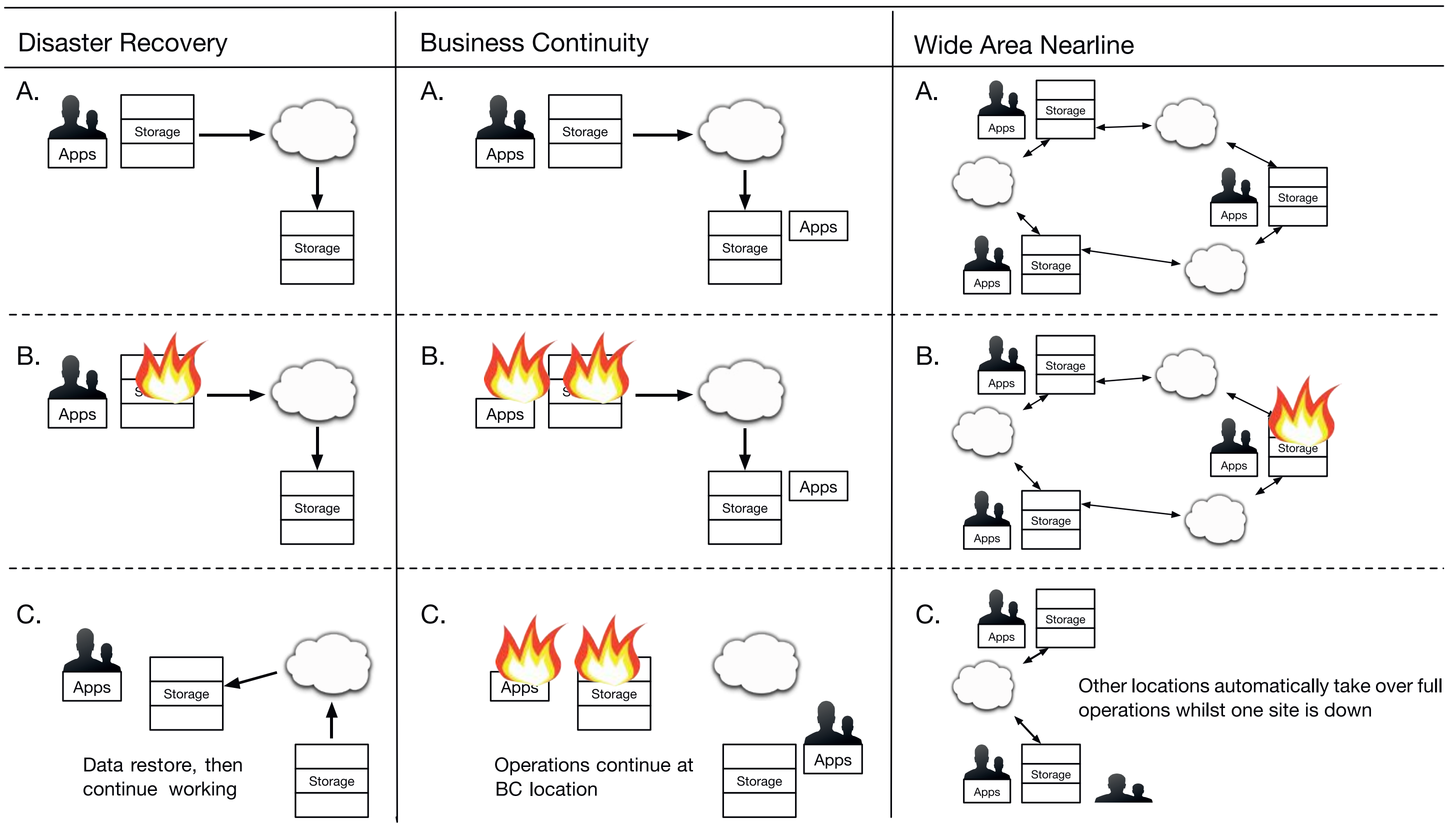
Disaster Recovery
Business Continuity
Wide Area Nearline
A.
B.
C.
Apps
Storage
Data restore, then continue working
Operations continue at BC location
Other locations automatically take over full operations whilst one site is down

Figure 1: Disaster Recovery vs Business Continuity vs Wide Area Nearline

Synchronous Replication vs Asynchronous Replication

Synchronous Replication means that the remote copy of the data exactly matches the local copy of data but whilst that sounds good on paper it is rarely employed as a solution due to the cost.

Effectively synchronous replication requires that the communications link to the remote site is so fast and reliable that the user of the data isn't affected by actions such as: the time lag required to disperse the data; flushing data to disk; locking the file; adding metadata etc.

If the second instance of the data is in the local network then speed of communication might allow for those types of actions, e.g., MatrixStore keeps two instances of data synchronised and linked within a single cluster.

However, over a wide-area-network things can slow down dramatically. And worse still, having synchronous long distance replication may make the local storage less reliable since any fault in the long distance communication may affect the timing and reliability.

Therefore, most people opt for Asynchronous Replication. Asynchronous replication sends data (and metadata) added or amended on the local system to the remote system, but there is a time delay between when those actions occur. This effectively *"decouples"* the storage of the data on the two systems and the local storage will carry on working without waiting for the remote storage to catch up. The price paid for this is that the remote storage will not be entirely in sync with the local system. This is generally seen as an acceptable compromise for most people akin to *"losing ten minutes of work is manageable"* as opposed to potentially losing a day's work or worse if all that is otherwise occurring is an overnight backup.

MatrixStore Replication Topologies Supported[1]

Internal to a single cluster MatrixStore keeps two instances of data, but where replication to second cluster is concerned MatrixStore supports asynchronous data replication only.

Any vault can be set to replicate to another vault on another cluster. This allows for the topology examples shown below, along with many other combinations that are not shown but can easily be imagined starting from the examples given.

When a vault is set to replicate then the items replicated are:

Description	Replicated?
Object data	Replicated
Object metadata	Replicated (though some system metadata may differ, such as last accessed time)
Vault Users	Not replicated: the destination vault will have its own user set. Separately to individual vault replication, the local cluster user set can optionally be shared with the remote cluster.
Object IDs	Although the object will get a new instance and consequentially a new object ID in the remote cluster, the object replicated can also be accessed and acted upon via its original ID.
Audits	Audits are not replicated.

Topology Example 1: Basic Vault to Vault Replication

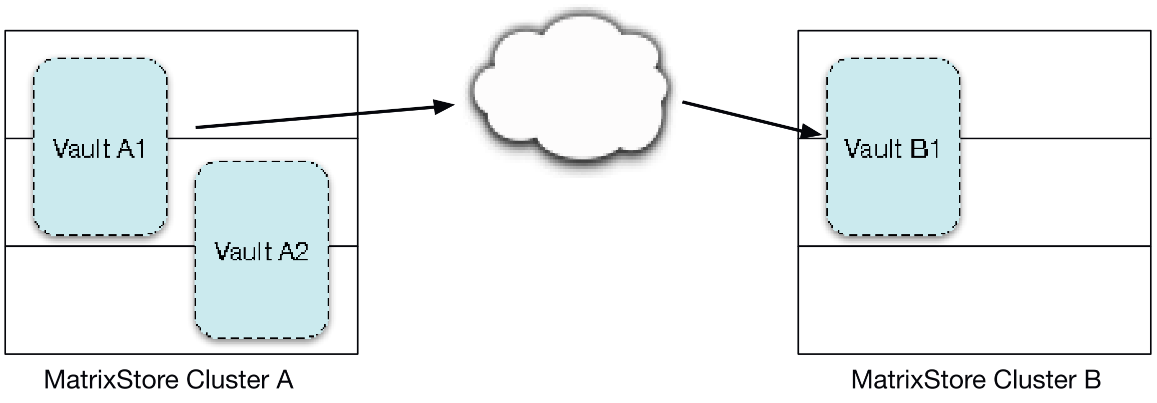

Figure 2: Basic Vault Replication

[1] *MatrixStore supports synchronous data mirroring within a single cluster*

In this example MatrixStore Vault A1 on cluster A has been set up to replicate to Vault B1 on cluster B. Vault B1 can also have objects added / updated / deleted locally.

Topology Example 2: Multi Hub to Central Repository Replication

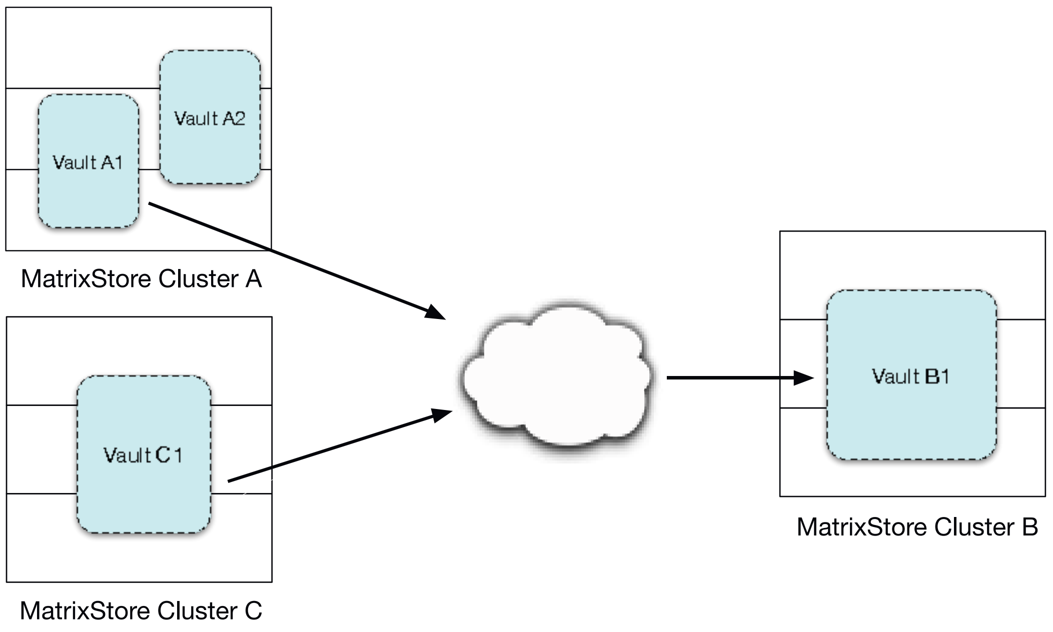

Figure 3: Central Repository Vault Replication

In this example MatrixStore Vault A1 on cluster A and Vault C1 on Cluster C have both been set to replicate to the same vault, Vault B1 on Cluster B. Vault B1 will effectively be a collection of all the objects from A1 and C1. This could be useful, e.g., in a situation where an organization has multiple satellite offices and wishes to collect all of the data into a single vault. Other options include: additional other vaults (e.g., in further clusters) sending data to the central repository; many vaults in the central repository – e.g., one per satellite hub, etc.

Topology Example 3: Multi-Satellite Forwarding

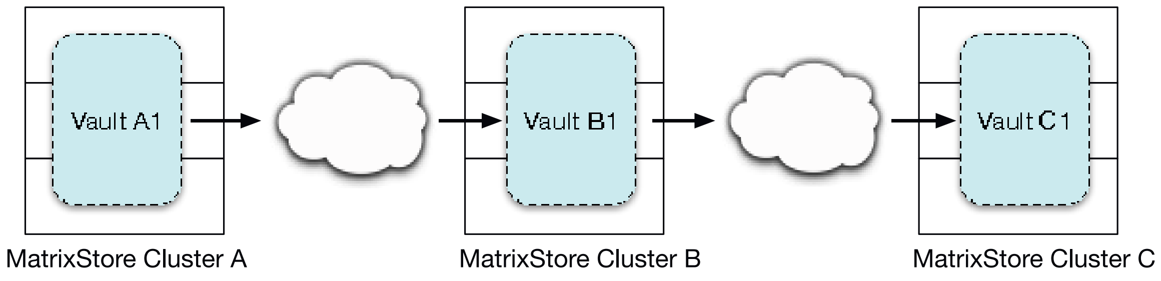

Figure 4: Multi-Satellite Forwarding

In this example satellite MatrixStores forward to one another and finally to a 3rd destination. In this way any data generated at Cluster A can be placed onto to every cluster. Similarly, see Topology 4 for how any data generated on any cluster can be seen at any location.

Topology Example 4: Multi-Replications

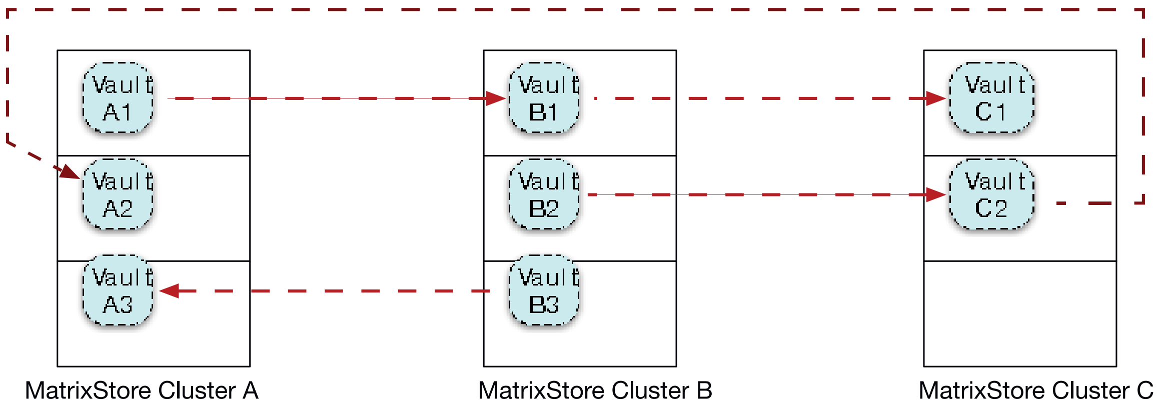

Figure 5: Complex Replication Setup

In this example:

Vault	Forwards to:
A1	B1
B1	C1
B2	C2
C2	A2
B3	A3

The point in this example topology is that everything put into vault A1 is available on clusters B and C. Everything put into vault B2 is available on all three clusters as well. Assets put into vault B3 are set to be available on cluster A vault A3. Sharing data across multi-locations is both easy to set-up and allows for flexible data distribution.

Two-Way Data Replication

Although the previous example's topology shows two-way data replication via using different vaults sometimes it is desirable to have two-way replication within the same vault. However, this is problematic in an asynchronous workflow:

- What if someone at cluster A updates an object at the same time as someone at cluster B updates the same object. Which version of the object is kept?
- The same problem applies to groups of objects: e.g., imagine a folder containing a media project. Imagine someone at cluster A adds an asset to that folder at the same time as someone at cluster B adds another asset. Furthermore, the project file is updated to use that asset. Now, after the objects are replicated, we have a strong possibility of an inconsistent set of data.

Although strategies can be employed for overcoming certain aspects of the above problems, these complications are the reasons why Object Matrix does not support two-way replication within the same vault.

Advanced Replication Concepts

Forwarding Deletes

MatrixStore provides the option to *"forward deletes"* or *"do not forward deletes"* in replication. If deletes are forwarded then once the object has been deleted on the first cluster, the second cluster will likewise delete the object. However, sometimes it can be desirable to collect a large pool of data in a central repository – e.g., if the central cluster is larger than the original cluster. If deletes are set "do not forward" then the object on the remote cluster will be left untouched by a local delete of that object.

MatrixStore Replication Mechanics

Under the hoods, one advantage of an object storage solution is that MatrixStore knows exactly which object's data or metadata have been added, updated or deleted.

Every node in MatrixStore is responsible for its own data and runs a *"Replication Task"* that looks at what data has changed and where data is to be replicated to whenever a change of data is registered. The task then uses the MatrixStore API to connect to the remote cluster and to send the data. MatrixStore can be configured so that every node simultaneously replicates to the remote cluster its data (giving a very wide aggregate bandwidth) but in reality most data tunnels between the source and destination clusters have a relatively limited bandwidth, therefore by default MatrixStore is set to only replicate from one node at a time.

The MatrixStore API automatically tries resending the data when a failure occurs. Optionally, all data can be sent encrypted. The API also ensures that the remote cluster has received the data bitwise correct.

When replication does go wrong (e.g., a bad communications line interrupts replication) then partial blobs may exist at the destination cluster. These can become sizeable if the communications line is very intermittent but can be cleaned up with the *"empty trash"* task in the admin tool.

There are options to keep zero instances of the object at a local cluster and to only keep the instances at the remote cluster. In this case a stub can be kept at the local cluster so that the object can still be searched for and recovered on demand. Furthermore, optionally replication can be set so that an instance of the data can still be found at the local cluster for a period of time... e.g., keep the data local for the first week after which the data can only be found at the remote cluster.

After replication has occurred a local cluster can recover data from the remote cluster in the case of local failure. E.g.,

1. A vault is set for single instance locally and single instance remotely
2. Replication occurs
3. A node fails on the local cluster
4. The cluster can be requested (requires a manual step) to recover data from the other cluster.

Advanced Interplay Replication

With the various options presented advanced and complex replication schemes can be created. MatrixStore can also be used in conjunction with partner applications to further automate workflows: one such example is how MatrixStore replication can be combined with InterConnect (from Object Matrix) and GlooPort from Glookast Inc. Putting the applications a workflow was created wherein:

- At site A multiple Avid Interplays are watched and the assets checked in they are automatically archived into the MatrixStore.
- The assets arriving in the local MatrixStore are immediately replicated to a remote MatrixStore.
- The assets at the local MatrixStore are kept local for a period of a week: this way they are kept quickly available to the Interplay users in case a *"backup"* copy is required.
- Changes on the local assets (paths or deletions) are reflected on the local archive, replicated to the remote MatrixStore and automatically applied to the remote Interplay.
- Assets arriving at the remote MatrixStore are automatically checked in (just the low-res) to an Interplay engine at that site with their workspace paths etc maintained. In this manner should a local Interplay cease to work for some reason, the remote Interplay can be used to immediately continue work on the projects in hand.

This workflow is shown in Figure 6: Advanced Interplay Replication Setup.

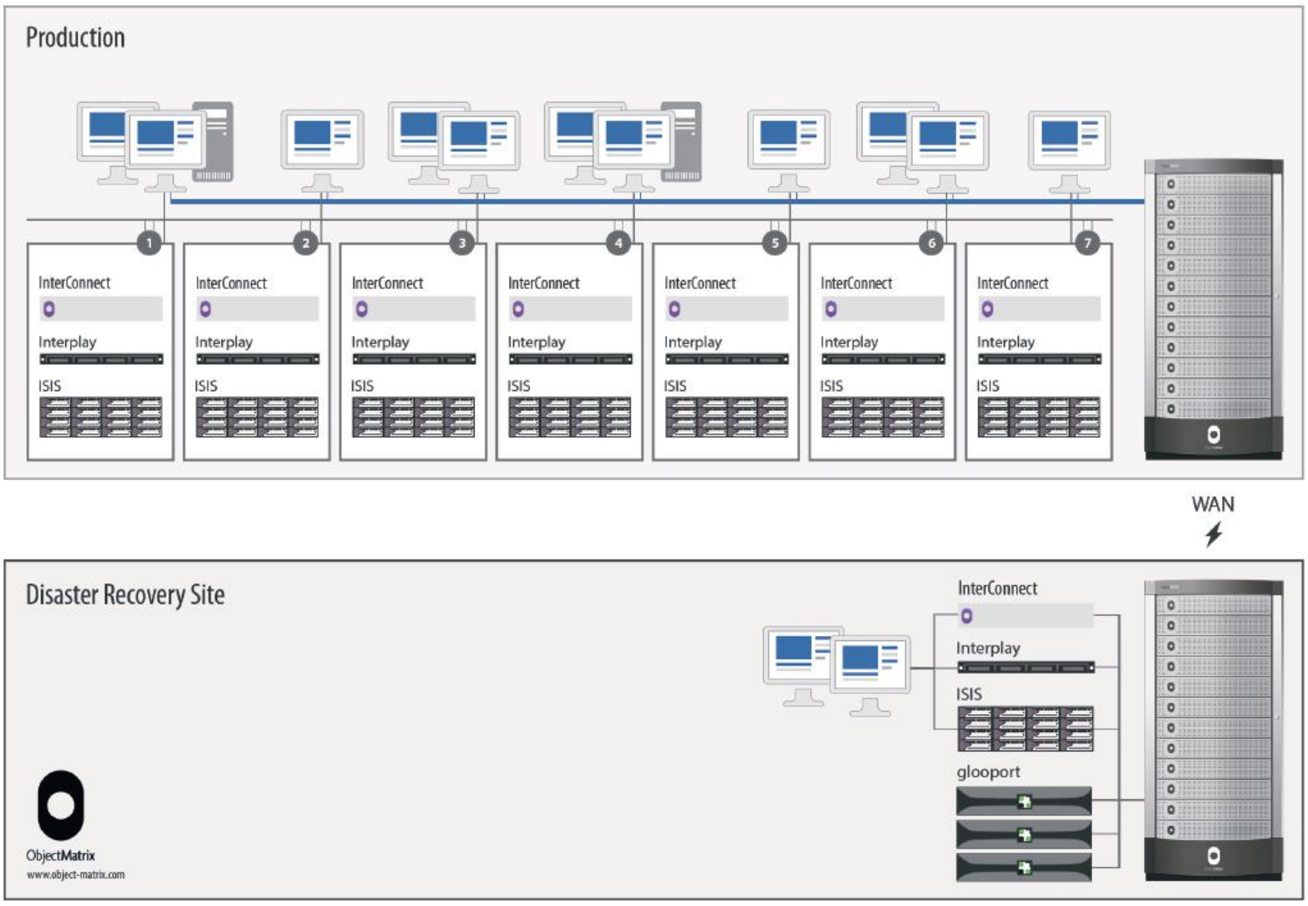

Figure 6: Advanced Interplay Replication Setup

Conclusion: Replication for Media Workflows

Whether the purpose is disaster recovery, business continuity or wide-are-nearline, replication of data forms the backbone of how data is shared between sites.

However there is no *“one size fits all”* replication model and subtle differences in replication techniques and topologies have a large affect on the outcomes and characteristics of the replication carried out.

That said, undoubtedly for many a major reason for using object storage is the ability to easily and effectively share data amongst more than one site and to be able to simultaneously use the resources of those sites for the production of media within the company. Simply having a second set of resources sitting idly by is a nice to have, but not feasible for all.

MatrixStore provides built-in a plethora of asynchronous replication options. By being built in it takes full advantage of the hardware architecture and intimately knows when an object or its metadata is changed (and therefore requires replication). Because the replication solution is built-in it is easy to implement and manage.
Advanced features such as being able to search on assets that are only stored remotely, or automatically recover data from the remote site are simply some of the icing on the cake.

About the Author

Jonathan Morgan researched Grid Computing at Texas Christian University in the 1990’s. In his work career he joined FilePool briefly before it was acquired by EMC for its object storage technology. That product went on to become *“Centera”*, one of the world’s first commercially successful object storage solutions. At EMC Jonathan led the largest development team for Centera developing *“content parity protection”* (an erasure code algorithm). Founding Object Matrix in 2003, Jonathan has been the CEO since its inception seeing the company grow from a concept to storing data for some of the world’s largest media organisations. Object Matrix is based in Cardiff, UK.