ObjectMatrix

Object Matrix
MatrixStore Architecture

Version 1.3.3, Dec 2014, Jonathan Morgan

ObjectMatrix

Contents

1 Overview

MatrixStore is the collective name for Object Matrix's software solution for disk based Nearline storage and archiving. This document describes the thinking and rational behind the product as well as the product itself. The document is written on a technical level for those who care about the nuts and bolts of why and how data is kept securely, forever, within MatrixStore.

It is the aim of this document:

- to explain key foundation "how-to" mantras for the solution
- to provide a deep understanding of the issues involved and why design decisions were taken
- to encourage discussion via understand and peer review to lead to design improvements and new functionality

2 MatrixStore Software Overview

MatrixStore software runs on a storage cluster made up of many "servers" commonly called "nodes". The software enables the storage cluster to act as a single entity such that connecting clients see the cluster as a single storage location.

MatrixStore is an object based storage solution. This means that each data element stored is treated as an object:

- A data element is typically a file, but could also be a piece of information such as a data record
- An object consists of the data itself, metadata to describe the data and policy control data

The art of a good object based storage solution is to provide not just storage of data but also services that aid the applications and clients connecting to the object store as well as inherent data storage capabilities such as high resilience to data loss or corruption and data storage policy controls. These are discussed later in this document in greater detail.

The software platform that makes up MatrixStore consists of:

Node Operating System:

A stripped down and optimised version of Ubuntu Server is used at the o/s level. This has been selected for its high reliability and ability to support various hardware configurations. XFS (file system) is used on each node within Ubuntu. Again, this was selected for its extremely strong track record, as well as its in-built resistance to fragmentation. MatrixStore Server software is largely independent of the o/s and has previously been run on various o/s including our own distribution of a highly secure Linux kernel.

MatrixStore Server:

MatrixStore Server auto-boots on each node running on top of the operating system.
Primarily written in Java, it occasionally calls on a few C libraries and on a few o/s scripts. Underlying fundamentals in the server layer are:

- Self-sufficient: each node is able to run without a dependency on services from other nodes.
- Lightweight inter-node communication: by keeping communication between nodes to a minimum the cluster is highly-scalable (by avoiding inter-node traffic noise).
- Highly-reliable: nodes should be able to run for several years at a time without requirement for reboot. This has been witnessed in real-world usage.

Architecturally the server is split into two layers, a “system layer” and a “services” layer. System layer is a minimal layer that is intended to be fairly static, provide security protocols, and to gather hardware information. Services layer handles all high-level cluster functionality, client connections, etc.

MatrixStore Clients:

Client connectivity to the cluster is always through one of two APIs:

- MatrixStore API: C or Java. This API allows connectivity for data storage, retrieval, search etc operations. This API has been programmed to by numerous 3rd companies.
- MatrixStore Management API: Java. This API allows higher level operations such as user creation and statistic information gathering. Historically this API has only been used by Object Matrix, however, in the future it is anticipated opening up the API for 3rd party companies to program to.

Clients can be on MacOS, Windows, Linux, some flavours of Solaris and UNIX. Clients communicate with the server using user datagram protocol (UDP) over TCP/IP.

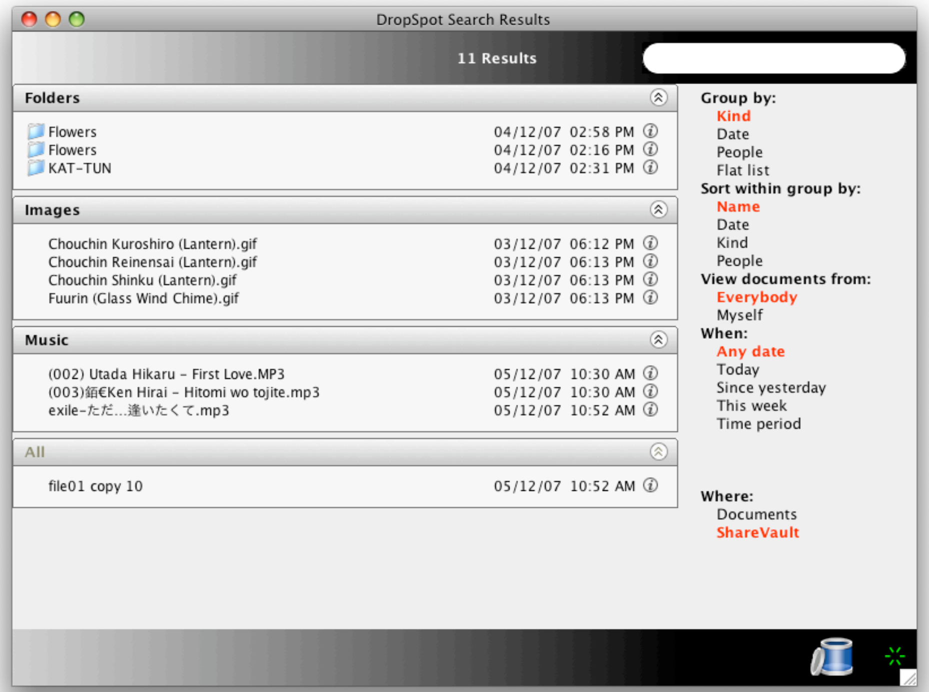
The MatrixStore APIs guarantees security, cluster location discovery, retries and much more besides.

Client Applications:

The following applications are provided by Object Matrix:

FTPConnect: Mac, Windows. This is an FTP server that runs on Windows or MacOSX. It translates client FTP requests into MatrixStore API calls to store/retrieve data in the cluster.

DropSpot: Mac, Windows, Linux. A client GUI application for performing data archive / retrieval / search. DropSpot can also perform these functions via shell commands. DropSpot is multithreaded and provides strong job control and verification - it is typically used for data ingest, e.g., camera data on to the storage.


DropSpot Search Results

11 Results

Folders	
Flowers	04/12/07 02:58 PM
Flowers	04/12/07 02:16 PM
KAT-TUN	04/12/07 02:31 PM

Images	
Chouchin Kuroshiro (Lantern).gif	03/12/07 06:12 PM
Chouchin Reinensai (Lantern).gif	03/12/07 06:13 PM
Chouchin Shinku (Lantern).gif	03/12/07 06:13 PM
Fuurin (Glass Wind Chime).gif	03/12/07 06:13 PM

Music	
(002) Utada Hikaru – First Love.MP3	05/12/07 10:30 AM
(003)餌€Ken Hirai – Hitomi wo tojite.mp3	05/12/07 10:30 AM
exile–ただ…逢いたくて.mp3	05/12/07 10:52 AM

All	
file01 copy 10	05/12/07 10:52 AM

Group by:
Kind
Date
People
Flat list

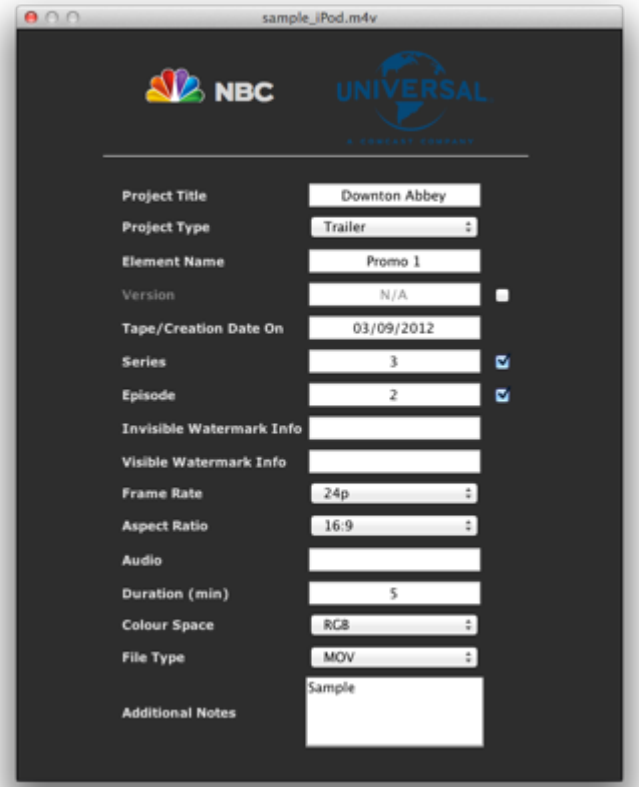
Sort within group by:
Name
Date
Kind
People

View documents from:
Everybody
Myself

When:
Any date
Today
Since yesterday
This week
Time period

Where:
Documents
ShareVault

Figure 1 - DropSpot main interface

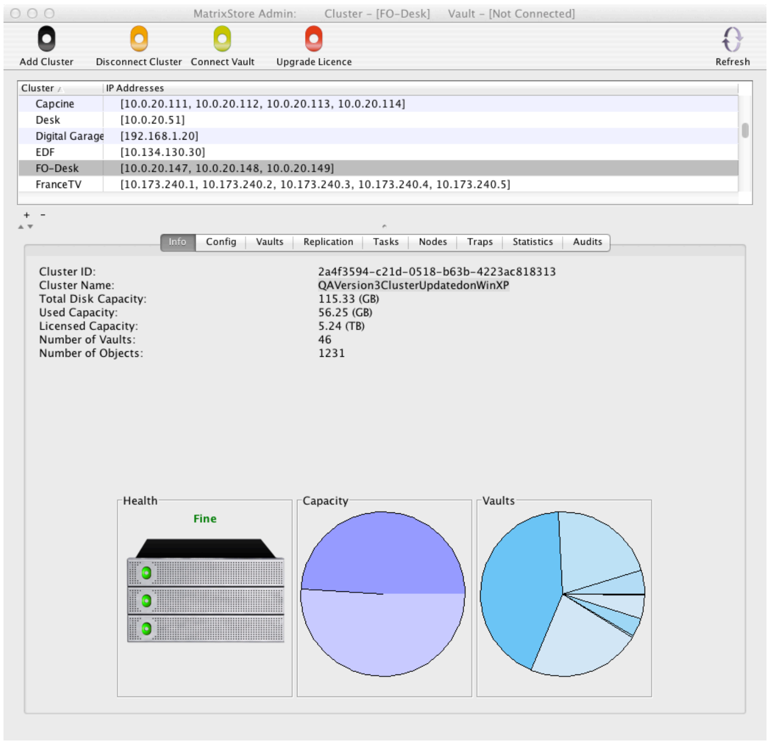


Project Title	Downton Abbey	
Project Type	Trailer	
Element Name	Promo 1	
Version	N/A	☐
Tape/Creation Date On	03/09/2012	
Series	3	☑
Episode	2	☑
Invisible Watermark Info		
Visible Watermark Info		
Frame Rate	24p	
Aspect Ratio	16:9	
Audio		
Duration (min)	5	
Colour Space	RGB	
File Type	MOV	
Additional Notes	Sample	

Figure 2 - DropSpot Metadata Form

MatrixStore Maintenance Tool: Mac, Windows. This application is typically used by Object Matrix engineers to create clusters, inject upgrades into nodes and to perform other low-level administration functions. One key feature of the maintenance tool is that it can simultaneously upgrade the software in an entire cluster rather than requiring nodes to be upgraded one by one.

MatrixStore Administrator Tool: Mac, Windows. This application is typically used by the administrator at the customer site to monitor a cluster and to perform basic administration upon the cluster.



Cluster	IP Addresses
Capcine	[10.0.20.111, 10.0.20.112, 10.0.20.113, 10.0.20.114]
Desk	[10.0.20.51]
Digital Garage	[192.168.1.20]
EDF	[10.134.130.30]
FO-Desk	[10.0.20.147, 10.0.20.148, 10.0.20.149]
FranceTV	[10.173.240.1, 10.173.240.2, 10.173.240.3, 10.173.240.4, 10.173.240.5]

Cluster ID:	2a4f3594-c21d-0518-b63b-4223ac818313
Cluster Name:	QAVersion3ClusterUpdatedonWinXP
Total Disk Capacity:	115.33 (GB)
Used Capacity:	56.25 (GB)
Licensed Capacity:	5.24 (TB)
Number of Vaults:	46
Number of Objects:	1231

Figure 3 - Admin Info pane

MatrixStore shell: Mac. This is only available to Object Matrix support and allows shell access to the MatrixStore cluster with a command layer. Shell access allows the support engineer to perform low-level engineering tasks.

MXFS: Mac, Windows, Linux. This filesystem can be launched on a client machine to provide file system type access to the cluster from that machine. The client machine will see the cluster as a drive letter / Volume and can subsequently share the drive to connecting clients. In the background MXFS sends and receives data to/from the cluster using MatrixStore API calls.

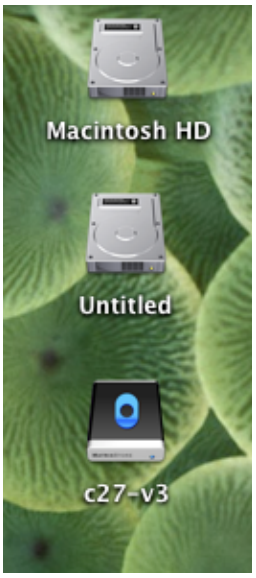
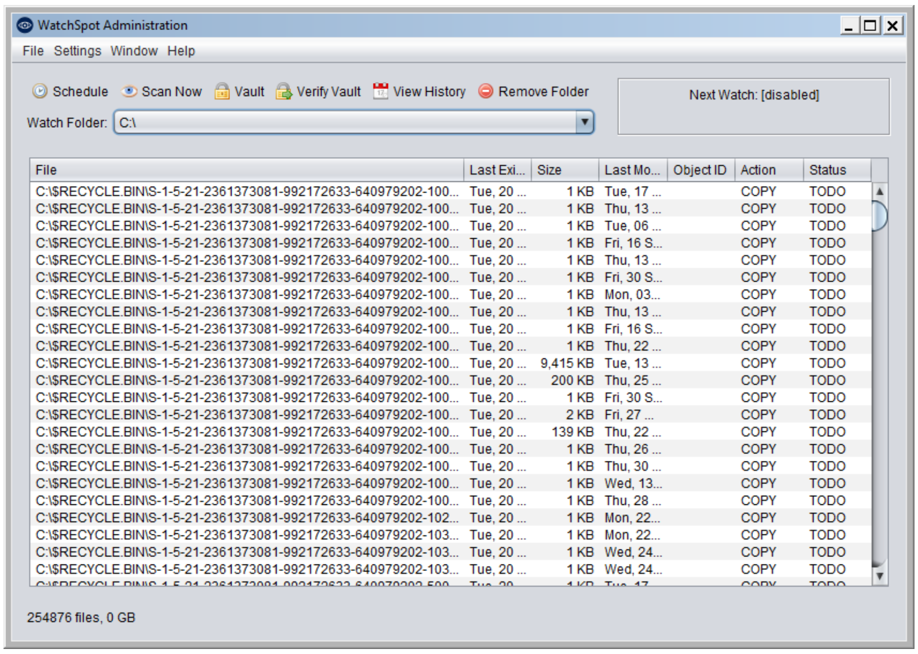


Figure 4 - vault c27-v3 mounted with MXFS

MXFS Samba Server: Linux. Launches a Samba Server v3 which reads/writes data to MatrixStore.

WatchSpot: Windows. WatchSpot is used to watch a folder on a filesystem and to sync the data on that folder to MatrixStore, or to watch a vault on MatrixStore and to archive data from that vault to the LTO service.



Next Watch: [disabled]

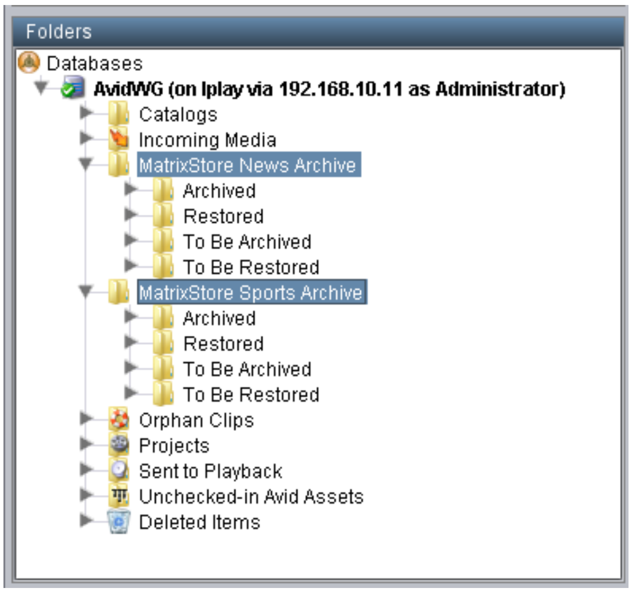
Watch Folder: C:\

File	Last Exi...	Size	Last Mo...	Object ID	Action	Status
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Tue, 17 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 13 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Tue, 06 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Fri, 16 S...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 13 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Fri, 30 S...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Mon, 03...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 13 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Fri, 16 S...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 22 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	9,415 KB	Tue, 13 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	200 KB	Thu, 25 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Fri, 30 S...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	2 KB	Fri, 27 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	139 KB	Thu, 22 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 26 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 30 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Wed, 13...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-100...	Tue, 20 ...	1 KB	Thu, 28 ...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-102...	Tue, 20 ...	1 KB	Mon, 22...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-103...	Tue, 20 ...	1 KB	Mon, 22...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-103...	Tue, 20 ...	1 KB	Wed, 24...		COPY	TODO
C:\$RECYCLE.BIN\S-1-5-21-2361373081-992172633-640979202-103...	Tue, 20 ...	1 KB	Wed, 24...		COPY	TODO

254876 files, 0 GB

Figure 5 - WatchSpot main interface

InterConnect: Windows. InterConnect will watch an Avid Interplay server and will archive user selected data from Avid Interplay to MatrixStore.



Folders

- Databases
  - AvidWG (on Iplay via 192.168.10.11 as Administrator)
    - Catalogs
    - Incoming Media
    - MatrixStore News Archive
      - Archived
      - Restored
      - To Be Archived
      - To Be Restored
    - MatrixStore Sports Archive
      - Archived
      - Restored
      - To Be Archived
      - To Be Restored
    - Orphan Clips
    - Projects
    - Sent to Playback
    - Unchecked-in Avid Assets
    - Deleted Items

Figure 6 - Interplay Access Folder pane

CatDV and FCServer Plugins: Plugins for 3rd party asset managers.

Move2 (move2SGL, move2Xen, move2ODA): These applications watch MatrixStore Vaults and manage the movement of data between MatrixStore and archive type storage devices. The applications can move data on demand from a user or according to business rules (age etc). When data is moved to a archive device, such as LTO, then a searchable stub is left in MatrixStore. The user, on discovering an asset, e.g., via DropSpot, can then elect to restore the asset within DropSpot. DropSpot will then call upon move2 to return the asset.

3rd Party Applications: There are many Object Matrix partners who have integrated their products into MatixStore as a storage solution.

3 MatrixStore Hardware Overview

Hardware from any vendor that supports Linux can be used with MatrixStore, although only hardware that Object Matrix has qualified is will be supported. A cluster consists of nodes and switches:

Cluster	Notes
Nodes	Minimum 3 (see below)
1Gb or 10Gb switches	Two for internal traffic

Hardware is grouped into nodes:

Node	Notes
CPU	Dual core 1.6GHz and higher
Storage	Up to 2 volumes, tested up to 192TB per node. On Enterprise nodes this is with RAID6.
Gigabit NICs or 10 Gigabit NICs	Two for internal traffic and two for external. External can be 1G copper RJ45 BaseT, 10G copper RJ45 BaseT or 10G SFP+ copper.
IPMI	One port for IPMI node control (optional)
OS	Ubuntu Linux (Server version)
RAID control	Hardware based for performance and reliability. Capacitor and flash backed RAM.
Memory	2GB or more (typical 8GB per node installed)

Typically Object Matrix supplies only enterprise quality components, such as Hitachi Ultrastar HDDs.

Points to consider in the configuration are:

- When a data object is stored into the cluster it is always initially stored to 2 separate nodes in the cluster. Therefore, a cluster must have at least 2 nodes in operation to be able to store data. Object Matrix therefore insists on a minimum of 3 nodes in a cluster to allow for data writes to continue in the eventuality of a node being down.

- If a node goes down, data is regenerated from the good node (that is still up) to a new node location. This ensures that within an allowable time period data is always kept with two good instances.
- For failover there should be 2 external traffic and 2 internal traffic NICS.
- Storage volumes are typically set to be RAID6. With the data kept in two locations (both RAID6), a minimum of 6 disks would need to irrecoverably fail for data loss to occur, furthermore, that failure would have to occur before data has a chance to regenerate.
- Nodes are typically configured with mirrored system drives. System drives are used to hold the operating system, system logs and some data that is shared between multiple nodes.
- RAM space of 1GB + 1GB per 20 million metadata entries (per node). The MatrixStore uses a 8GB configuration.
- CPU speed: Recommended to use at least dual core 1.6GHz, preferably higher.
- Cluster is designed for up to 80 nodes (15PB), however, there are no hard and fast reason not to go above that number. Cluster is tested up to 40 nodes.
- Time for the cluster to self-heal is a calculation based upon the size of the node that has gone down ÷ the number of nodes in the cluster ÷ internal NIC speeds. Or: the smaller the size of each node and/or the more nodes there are, the faster regeneration will be.
- The more nodes there are the greater the bandwidth of Ethernet connections to external connections to the cluster. A “fast” cluster should therefore have low amounts of storage per node, a “high density” cluster should have high amounts of storage.

Networking

Client machines access the cluster via IP addresses. Nodes within the cluster contact each other via the internal network.

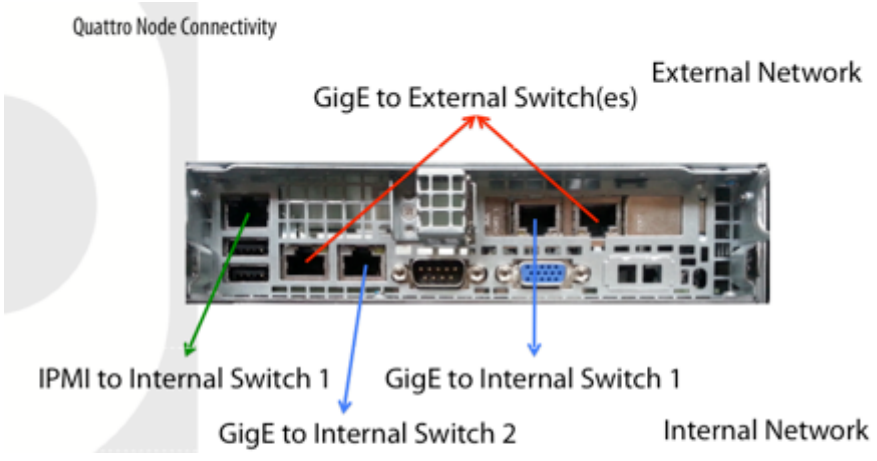

Quattro Node Connectivity
External Network
GigE to External Switch(es)
IPMI to Internal Switch 1
GigE to Internal Switch 1
GigE to Internal Switch 2
Internal Network

Figure: Typical node’s physical network configuration

External traffic	Examples
API, ports 1907, 1908	Write and read operations
Management API, port 667	Vault creation, user modifications
Secure Socket Layer (SSL), port 8443	

Internal traffic	Examples
Management API, ports 666, 667	Vault creation, user modifications
API, ports 1907, 1908	Write and read operations

External maintenance traffic*	Examples
SSH, port as allocated	ssh should normally be switched off.

* Generally these ports can be blocked on the external network

When configuring a MatrixStore it is essential to:

1. Assign an internal and external IP address to each node
2. Ensure that firewalls within the network do not block traffic on any of the above external or internal ports*
3. Ensure that each node can see each other node via Ethernet networking, both within the internal and the external networks

When configuring MatrixStore it is normal/strongly recommended to:

1. Configure one (or two for failover) network interface ports to be dedicated to internal traffic
2. Configure one (or two for failover) network interface ports to be dedicated to external traffic
3. Dedicate one (or two for failover) switches to internal traffic, and thus isolate internal traffic from the rest of your organisation’s network

Switches will typically be unmanaged Ethernet. As standard, Object Matrix implements failover on the switching using switches that support IEEE 802.3ad.

Data transmitted over the internal network is not encrypted. Therefore, internal switches should be physically isolated and protected. Because the internal switches sit behind the nodes, compatibility with switches in the rest of the client’s network is not required.

Port connectivity to the cluster from a client can be checked through the menu option Cluster -> Test Connection in the MatrixStore Administration Tool.

4 MatrixStore Concepts

4.1 Overview

MatrixStore is an object based storage solution. Every object based storage solution has strengths and weaknesses that are a result of underlying architectural decisions and project aims. MatrixStore includes the following high-level ambitions:

- Scalable solution to 100 Petabytes; not, scalable to Exabytes (e.g., Amazon S3)
- 100's of simultaneous users; not, millions
- Hybrid solution usage: fast object storage plus full speed filesystem access; not limited to either access type. This requires exceptionally fast random access to objects.
- Provide highest levels of data security and data protection
- Completely plug and play architecture
- Object storage via data + metadata + data storage policy control
- Full suite of data services including search, metadata handling, replication options, self healing and full support for an eco-system of client applications
- Capable of mixing different hardware nodes, e.g., as a result of scaling the cluster over a number of years

Why archive to disk?

Disk is a proven well-understood technology with good transfer speed and excellent random access. File systems are also well established. However, in terms of archive for large organisations the challenge comes when 1000s of disks need to be managed as storage locations, including the challenges of authenticating that data is maintained bitwise exact.

Normally on disk based solutions:

- Large scale disk solutions are difficult to manage, e.g., 100TB+ SAN solutions
- Bad administration can very easily lead to malicious or accidental data loss
- Solutions become outdated and data needs to be transferred to the "next generation" solution
- Downtime is caused by individual component failures
- Transferring data offsite, search, firewalling, etc., requires separate software modules, each of which might result in data loss and downtime when upgraded or broken. Over time, the systems become disjointed as individual components become out of date.

MatrixStore seeks to overcome each of those problems.

Disk based archiving is a proven way of keeping large amounts of data:

- Google use it as the basis for keeping their data
- EMC Centera clusters data on disk and applies rules to the data such that it complies with many regulations
- Isilon clusters data on disk for a “SAN” type solution, placing a Samba type interface in front of the storage

Etc. In fact, if cost/technology was not an issue, then mostly anyone would like to keep their data online, available, with faster access and more easily managed on disk rather than offline on tape/optical.

4.2 MatrixStore Data Services

MatrixStore software is server software that is installed onto each node in a cluster. It runs 24/7 to monitor the cluster and to perform services such as data organisation, search, self-healing, replication of data, to apply business rules to data handling, to handle client connections for write and read and to generally virtualise the hardware and disks within a cluster to appear as a single storage location.

MatrixStore software is designed from the inside out to support the requirements of Nearline storage and medium to long-term archive.

MatrixStore makes the cluster:

- secure / ‘firewalled’
- near zero maintenance, e.g., to scale/to manage large volumes of data
- provides guarantees about the delivery and storage of data to and from the disk
- monitors the archive for hardware failure, taking automated actions where required
- fast
- searchable
- scalable
- functional
- easy to roll-in/roll-out technologies into the pool of storage as time goes on

4.3 Object Based Storage

It wasn’t so long ago that just about all data was stored within a file system, but as the scale of data storage has increased dramatically, so the rise of object based storage has become an essential part of keeping digital assets. Examples of object based storage include Amazon s3, Google, EMC Centera, to name but a few systems. Object based storage has the following fundamentals:

- Like files, objects contain data, but unlike files, objects are not organized in a hierarchy. Every object exists at the same level in a flat address space (sometimes called a storage pool).
- Objects can also represent records rather than data files.
- Both files and objects have metadata associated with the data they contain, but objects are characterized by their extended metadata. Each object is assigned a unique identifier that allows the server or the end user to retrieve the object without needing to know the physical location of the data. This approach is useful for automating and streamlining data storage in scalable environments.
- Within MatrixStore, the user need never know the unique identifier, since the metadata can be used to look up the object.
- Objects are self-describing: unlike in a file system where should metadata servers blocks of data are individually meaningless, object based solutions are highly resilient to data loss through their very atomicity.

On top of massive pools of objects virtual views can be built to view the data. One view of a bunch of objects might be from an asset manager, another view might be a file system view. However, in order to support file system views the underlying object based storage solution must support expected file system behaviours, such as response times.

Under the hoods, a MatrixStore cluster actually stores objects in numerous file systems. Benefits of this are:

- Proven technology
- Low interdependency between objects
- Scalable over main nodes.

Negatives:

- Files have an overhead of 200 bytes; unfilled blocks may result in wasted space; therefore MatrixStore is not suitable for efficiently (in terms of overhead in disk used) storing millions of very small (sub 100 byte) objects.

MatrixStore is designed for the storage of millions of objects.

When data is stored into MatrixStore it is stored as an “Object”. An Object instance consists of the original data in an unchanged format and an associated metadata file containing user and system attributes for the object.

All data is stored into virtual constructs called “vaults”. A vault contains meta attributes pertaining to the handling of objects within that vault, such as whether those objects must comply to regulations (See “Enforcing the longevity of data”) or whether data should be replicated, etc. In effect, that set of policies can be considered to be attached to each object.

4.4 Object Metadata

Object metadata is used by MatrixStore and by users for different purposes. Users generally use metadata to identify objects, but may also use metadata to hold attributes about the object.

The MatrixStore server uses metadata to perform system tasks, such as to authenticate that a corruption hasn't occurred within the data file.

Object metadata is primarily stored in two locations:

1. in "flat file" format along with the object itself
2. in a searchable database

The "flat files" are stable and are generally stored alongside the data files. The contents of the flat file is a standard Java HashMap, wherein the keys are ASCII-US text, and the data is the binary characters representing the value of the attributes. Since each object stores its own metadata in its own file, corruption in one will not affect another, however, the downsides are the filesystem overhead of storing extra files as well as storage speeds if many attribute updates are made to an object. It is not expected that rapidly changing attributes (e.g., a counter) will be stored within MatrixStore objects.

Every node stores a database of all of the metadata upon that node. This allows for distribution (as clusters grow) and low levels of interdependency between nodes. Should the database become corrupt, it can be rebuilt from the metadata files on the node.

Metadata can be attached with two types of identified to an object:

- System attributes are used internally and are not commonly returned to the user.
- User attributes are those added by the user and/or 3rd party applications (e.g., CatDV)

When stored, attributes can be tagged as searchable or non-searchable. Only searchable attributes are added to the server database.

MatrixStore (v3.1) also extracts metadata (and can extrapolate additional metadata) from the contents of objects.

4.5 Vaults

Any object stored in MatrixStore is in fact stored into a logical entity called a vault. As many vaults as required can be created in a single cluster (Object Matrix tests for up to 2000 vaults per cluster, but can test for higher numbers of vaults should there be a requirement to do so). Vaults are intended to be used:

- One per project
- One per department
- One per data type
- One per customer

Vaults are not normally intended to be used per user, rather, per user group.

When a vault is created it can be given a set of properties dependent on the data to be stored within it. Those attributes are:

- Provisioned Capacity (see section below)
- Object Policies
    - Replication settings
    - Group access settings
    - Compliance settings (data retention period, retention policies)
    - Audit settings

Via vaults, MatrixStore supports "Multiple Tenancy" and can provide limited statistics for companies wishing to implement chargeback mechanisms for vault throughput / storage space usage.

Vaults are created / modified via the Management API. This is called from the MatrixStore Admin application.

Security between vaults is stringent (see Security section). Therefore:

- Using credentials from one vault user will never allow you to read/write/etc to another vault
- The system administrator is not allowed to write / read / etc data within a vault, but he does have permission to reset the vault administrator's account
- A vault administrator can (optionally) be allowed to write / read / etc data in a vault, but he does have permission to create / reset vault users

4.5.1 Vault Provisioned Capacity

It is possible to set and update a boundary on the capacity that a vault can consume within the cluster.

Should the provisioned capacity be exceeded the users of the vault will not be able to perform any further write operations until such time as the provisioned capacity is extended or the user deletes existing data in their vault thus freeing up capacity.

It is also possible to set the provisioning to be boundless, as such a vault can grow and grow as long as there is cluster capacity available.

Capacity is not reserved for the vault, thus, many vaults could be given a "1000TB" provision, even if only "100TB" of storage space is available on the cluster.

The system will allow writes until that capacity has been exceeded. i.e., if the last object written is a very large object, it could be that the capacity is significantly exceeded, but then the next object write will fail.

4.5.2 Object Longevity Guarantees

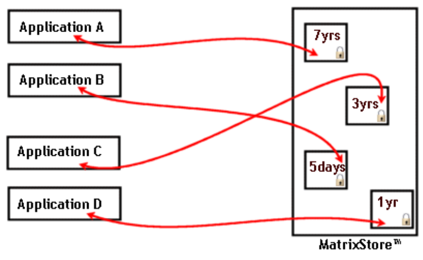
When designing MatrixStore to hold archive data, it was seen as essential to allow users to lock down data such that it cannot be deleted for a period of time or even forever.

Locked data is effectively read-only for a fixed (or administrator adjustable) amount of time.

This can be useful for a multitude of reasons:

- to protect against viruses/malicious users
- to protect against human error (oops! Just deleted the data in the wrong vault type errors)
- to comply to a multitude of government and/or industry regulation compliance requirements

When the user creates a vault the vault object is set to have a compliance attribute. The compliance attribute can be set to be on or off. If on, the amount of time that the data is stored for can be modifiable or extendable only.



When compliance is on, any object stored cannot be deleted for the period of time indicated in the vault.

If the setting is extendable only then the length of time that was selected can only be increased (not decreased). This means that the administrator cannot quickly change the setting, delete a file and then change the setting back again.

If the setting is modifiable, then an administrator can change the setting to then delete a file.

Notes: Although the "modifiable" setting doesn't guarantee data longevity against user actions, it does however guarantee that any non-system administrator will not delete data, and is therefore popular in smaller companies.

Data will not automatically be deleted when the longevity date / compliance threshold has expired

MatrixStore allows metadata pertaining to objects to be modified at any time.

MatrixStore implements the compliance setting at a vault level. When a user wishes to perform an action such as a "delete" the settings of the vault are checked, by the MatrixStore Services layer, before allowing the delete to occur. This effectively stops users without physical access to hardware from being able to delete objects that have been stored protected.

4.6 MatrixStore Regulation Compliance

Government regulations often include requirements/ guarantees to be made about the way in which data is handled, accessed, security protected, protected against loss, protected against accidental and/or purposeful deletion, searched, stored, audited and authenticated.

MatrixStore helps to support, amongst others, the:

- Security and Exchange Commissions rule Rule 17a.4 that aims to prevent overwriting, erasure or alteration of records
- HIPAA privacy ruling for Data Protection, requiring compliant backup methodologies to ensure the security and confidentiality of patient records
- The Sarbanes-Oxley Act of 2002 protecting investors by improving the accuracy and reliability of corporate disclosures. The Act amends mail and wire fraud infractions with harsher punishments and imposes fines and prison sentences of up to 20 years for anyone who knowingly alters or destroys a record or document with the intent to obstruct an investigation.

The set-up of a vault should be governed by the classification or type of data to be stored in that vault in accordance with any internal or legislative requirements. Once set-up, data stored into that vault will be enforced to comply with that set-up.

MatrixStore achieves compliance via functionality implemented in the server layer and the client:

Data Immutability

- MatrixStore can lock down data and can be set to disallow any updating of objects archived

Data Longevity/Non-repudiation

- Vaults can be set such that data cannot be deleted for a period of time. The MatrixService layer enforces adherence to the policy. The period of time cannot be changed if the vault is set to be unchangeable
- Different vaults can be set for different data types

Data Security[1]

- Communication with the MatrixStore can be up to 128-bit asymmetrically encrypted
- A public key encryption mechanism is used such that data sniffing, data modification, and replay attacks are protected against
- Vaults can be created so that the system administrator does not have access to the data, only the vault users

Audits

- All actions on vaults/data may be audited
- Audits can be set to include/exclude read operations
- Audits are maintained, secured, and protected to the same levels that data objects are

Search

- Data may be searched back using metadata labels.

Trusted 3rd Party Verification

- Data may be replicated offsite, where appropriate, to a trusted 3rd party

Data Protection and Authenticity

- Data and hardware is automatically monitored to be correct, thus ensuring the long-term authenticity of data down to a bit level
- Replication of data to another site ensures protection against physical hardware attacks

Delete

- MatrixStore does not currently shred (overwrite deleted data multiple times with zeroes before deletion), but could easily changed to do so if that is a requirement for someone

MatrixStore has a very strong set of functionality to help meet compliance requirements.

4.7 Data Compression, Data de-duplication, Content Addressed Storage (CAS)

Data Compression is only significantly relevant to companies that store uncompressed data. Object Matrix takes the view that most of its customers are storing video format files, and recompressing these files would be both time

[1] Versions of MatrixStore that do not include such strong encrypted data options are available for territories with export restrictions.

consuming and fruitless. Furthermore, compressed files cannot easily be edited or partially restored. Client software can easily compress data before storing; MatrixStore does not do this automatically.

Data de-duplication is not currently a feature of the cluster but is often carried out by the storing client application (backup software, Final Cut Server etc). De-duplication comes with a strong drawback - that it creates data fragmentation. Object Matrix believes in storing data in an open, well-established formats.

Content addressed storage (CAS), if required, is easily implemented in MatrixStore, simply store the data's digest with the data object as an attribute and use that to search back the data when required. CAS has generally become an out-dated concept.

MatrixStore is setup to keep data in an uncomplicated, secure, and long-term fashion and has a healthy disrespect for risk.

4.8 MatrixStore Services

To allow nodes to act as a coherent group of loosely coupled, yet self-supporting and scalable architecture, server software runs on each node providing a set of services.

Internal services are covered in the Tasks section of this document. External services include some tasks and:

External Service	Notes
Node status	Communicates node status to rest of cluster
Multi-node transaction service	Perform transactions to data (e.g., add attribute), in a synchronized, guaranteed manner across nodes
SNMP service	Provide SNMP status

4.9 Search

MatrixStore is extremely efficient at searching for metadata since it takes advantage of distributed search across the nodes and an optimised (for metadata) database.

MatrixStore is designed to allow up to 20 million entries into a DB, per node, per 1GB of memory. A maximum of 8GB of memory can be used per node. A typical cluster can handle and respond to thousands of search requests per second with zero to low impact on other concurrently occurring operations.

Thus, since the database is distributed, it scales in capacity as the cluster scales upwards in nodes.

MatrixStore guarantees attributes about the database that are essential for long-term cluster integrity and low-maintenance:

1. That the database can be rebuilt from the objects stored. That is to say that the object's metadata is stored within each object as well as within the database.
2. That should a component part of the database become lost or corrupted, that the database as a whole can continue to function.
3. That the database does not become slower, or require "tuning" as the number of metadata items it holds grows.

Some organisations may require or intend to use high numbers of metadata items. The metrics around the number of metadata entries vs node storage capacity may have an affect on cluster configuration selections.

4.10 Object Encryption

MatrixStore does not automatically encrypt the data stored, but it can encrypt data during transmission. If encryption is required, then it should be carried out in the client software layer and should use a certificate authority whose lifespan will outlive the archive!

4.11 Object Writing, Load Balancing

All data written to MatrixStore is written through the MatrixStore API. The API ensures that the transmission is completed quickly, successful, and when selected, securely.

The steps involved in sending data are:

1. Client initiates the construction of a secure connection to the cluster. Connection can be to any node.
2. Request is sent to obtain a location to send the data to. Client is given a location and a security certificate to send the data.
3. Client sends data to that location using a direct IP link to the node.
4. As the cluster receives data at one location it simultaneously relays the data to a second storage location via the internal network.
5. When an end-of-object indicator is sent by the client to the cluster, client also sends the checksum for the object. MatrixStore confirms that both nodes have received the correct data with the same checksum, and then syncs the data to disk[2] together with any metadata and policy information. An unique ID for the object just stored is returned to the client.
6. The client can (if required) safely remove the original copy of data, knowing that the data has been correctly received and flushed to disk at two separate locations.

[2] In the case of capacitor and flash memory backed up disk cache, this may mean that the data now sits in the cache rather than on the disk itself

As can be seen by the steps above, secure, reliable transmission of data is paramount; checksums, secure transmission and careful multiple location sync'ing of data are not good for performance but are key to archiving.

Data transmission options are:

Type	Notes
Secure	Data is up to 256-bit encrypted when sent over the connection. Note that MatrixStore builds are also available for territories with export restrictions that include lighter or no encryption options. The packets are encrypted with their own keys, with the key being changed during each handshake with the cluster. Data transmission is therefore strongly protected against sniffing and replay attacks.
Unsecured	Only the connection to the cluster is secured. Unsecured transmission is faster than secured transmission, although the difference in speed depends on the hardware used (30% is typical).
Asynchronous	Either of the above options can be used in an asynchronous mode. Using this option the client continues to stream packets to the server without waiting for ACKs, but rather, checking that the server has received the correct data at checkpoints.

Load balancing is essentially carried out on a round-robin basis, though the algorithm can also takes the following factors into account:

Factor	Notes
Disk space remaining	Nodes with the most disk space remaining are preferred to those that are almost full.
Recent activity	To a lesser extent, load is spread so that the entire cluster is used (not just the most empty nodes)

Data will always be stored in two separate nodes. Thus, should a node go down, the data will be available for reading from the second location. The user can however select that a vault keeps only a single instance of data. If that is the case then once the data received has been verified (ie, read back from disk) then one instance of the data will be removed.

Note that since data is kept in just two locations if massive multiple user read access is required then a read cache should be used in front of the storage. This is typical in VoD (Video on Demand) type services where an individual film may be being watched by 1000's of end users.

When data is sent through the API to be stored, it is sent together with metadata and policy information. This information is stored together with the data to form an object instance.

Data is stored in a non-proprietary format on the disks.

4.12 Performance

Unlike filesystems which typically go through metadata controllers (which then become the bottleneck), via TCP/IP connections, one client writing data to one MatrixStore node will not affect the performance of another client writing data to another MatrixStore node.

On 1Ge, MatrixStore performance is at full gigabit Ethernet speed, and because individual clients connect to individual nodes to store data, as the number of connections and clients grows the cluster can store data with near linear speed improvements. 10Ge options are now also available.

For a single file transferring from a single client the following can bottle neck performance:

- Since an IP based protocol is used ACKs can limit performance on high latency networks. For internal networks this is not an issue.
- CPU speed at the server is critical when interpreting incoming packets and flushing them through to the disk buffer. Sub dual core 1.6GHz processors will struggle to handle 120MB/s.
- Disk RAID write speed can limit performance, particularly with 10Ge
- Network - a single 1Gb line will limit to max 100 to 110MB/s over IP

4.13 Object Reading

As with writing data to the MatrixStore, all read operations pass through the MatrixStore API. The MatrixStore API enforces that the connection is made with the correct security credentials and then that the transmission of data is both checksummed and secure.

The following steps take place during reading data:

1. Client initiates the construction of a secure connection to the cluster
2. Request is sent to obtain the locations from which to read the object from.
3. Client reads data from one of those locations using a direct IP link to the node.

The cluster load balances reads using a simple random selector to select from which node the client should read the data.

4.14 Object Identification

Every item of data stored is stored as an object. Every object is given an unique ID by the MatrixStore. The ID is a 256 bit unique identifier (in an ASCII printable format).

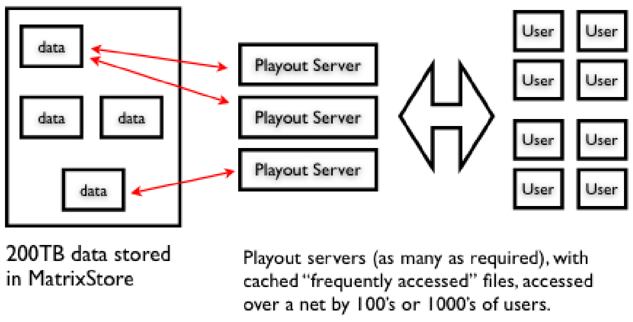
When an object is stored, and the ID returned, the client may select to store the ID in a database to be able to retrieve the object at a later date. However, it is not necessary to do so if enough metadata has been attached to the object to allow it to be identified via that means. E.g., the storer may wish to attach its own ID to the object and use that as the means to retrieve the object. Whilst requiring an extra round trip between the client and the server to start reading the object (the first round-trip being to perform a search, the second to retrieve the object), the high speed of searching means that this is still an excellent way to retrieve data.

4.15 Quality of Service

SNIA define QoS as “A technique for managing computer system resources such as bandwidth” … “Policy rules are used to describe the operation of network elements to make these guarantees.” … “RSVP allows for the reservation of bandwidth in advance”.

MatrixStore does not provide QoS electing rather to serve data as quickly as possible rather than to guarantee a steady stream of traffic.

MatrixStore should not be seen as a playout server, rather, it can provide the playout server with all the data it requires. For example, a data centre topology may have:


data
data
data
data
Playout Server
Playout Server
Playout Server
User
User
User
User
User
User
User
User
200TB data stored in MatrixStore
Playout servers (as many as required), with cached "frequently accessed" files, accessed over a net by 100's or 1000's of users.

4.16 Security Overview

From the inside out MatrixStore was built with security in mind:

- security is included “out of the box” without requirement for specialist training or knowledge;
- security is on the logins, data transfer, and on the node firewalls;
- security is simple: because the only access allowed is via the APIs any security hole would have to exploit those APIs which is highly controllable

Security - Firewall

The ports required by the MatrixStore are 666, 667, 1907, 1908, and 8443. They use protocols designed for MatrixStore only, and communications over these ports are fully secured.

The Linux OS build has been stripped down to remove all applications from the OS that could communicate over other ports. Since there is only a minimum of background processes are running, the node is stable as a reliable long-term storage device.

When installing MatrixStore, it is optional to leave an ssh port open on the cluster. If left on, the system is only as secure as the password(s) on that port. It is therefore recommended not to leave an ssh port open.

Note that if the company installing the MatrixStore has a firewall installed in the data centre, then it must be adjusted to allow for traffic on the ports used within MatrixStore.

Security - Data Communications[3]

Every client that talks to the server is uses MatrixStore's API in that connection (important: see footnote). The connection sets up a PPK type connection, the sequence of events being:

1. Request and agree up to a 256-bit secure channel to communicate over with the server
2. Pass a 256bit password over this channel for access to the vault
3. Transmit any data. Data is up to 256-bit encrypted.
4. On the handshake of each packet, agree the encryption code for the next packet.
5. On receipt of a packet, de-encrypt (against previous handshake)
6. (until end of communication)

Thus,

1. All communication is encrypted and authenticated
2. Replay attacks will not work (due to evolving keys)
3. Packet decryption over long time packet sniffing is difficult due to the usage of evolving keys
4. Data can be safely sent over the Internet, and indeed the whole cluster can safely be placed on the internet

To encrypt data the Helix algorithm[i] is used. The security solution has been built to allow other algorithm's to be plugged in as required.

Whilst it is desirable to encrypt the traffic of data to and from the server, the act of doing so generates CPU overhead at both client and server sides. In situations where data does not need to encrypted, encryption may be switched off.

Security - User Types

Data is stored and accessed within logical constructs called "Vaults". A client with access rights to a Vault has access to that and only that Vault (i.e., unlike groups in UNIX type systems). Thus, a client will need to maintain separate

[3] MatrixStore version for export to restricted territories does not include these options

access credentials if he/she wishes to access different Vaults on a MatrixStore archive.

Vaults can be organised as required, e.g., one vault could be for pre-production data, another for post-production data, another for customer-x, and yet another for the accounts department.

Access credentials for a Vault can provide read, write, delete and/or search capabilities for that vault. Thus, a customer could be given read only credentials for a vault relating to the customer.

There are several System level user types that have no access to data in Vaults but are allowed to perform different functionality via the MatrixStore Management API:

User Type	Notes
root	Generally this user is not used. Can create/change the sysadmin. Root password is created at cluster creation time.
sysadmin	Commonly used cluster password via the Administrator console. Can create vaults, reset vault passwords, change SNMP settings, etc. A sysadmin account is created when the cluster is created.
service	Designed for a service engineer. Provides the service engineer with access to only functionality that will aid the engineer, to avoid giving access to sensitive data.

Per vault users are:

Vault User	Notes
Admin	A vault admin may create users on the vault
user…	Vault users can be given read / write / delete / search rights.

Security - Viruses and Human Error

Most disk based archive or mass storage solutions are extremely vulnerable to viruses and/or human error. A virus with access to a volume could quite easily delete the content of an entire archive. Likewise, a human error, e.g., typing "rm -r *" at the wrong location, could easily lead to data loss. MatixStore is built to avoid such losses.

All communications to the cluster go via secure protocols. A virus would need to have intimate knowledge of the protocol and even then would need to have the right security credentials.

Furthermore, all data can be stored as immutable for a determined amount of time thus stopping data from being accidentally and/or deliberately deleted.

Security - Adding Nodes

To avoid rogue nodes trying to attach themselves to the cluster as a malicious entity or simply before the administrator is ready to start using the node, new node attachment to an existing cluster is a two stage process:

1. Install MatrixStore software on the new node, and physically attach the new node to the other nodes in the cluster.
2. Use the Maintenance tool from a client machine to add the node to the cluster. This requires a system administration password.

4.17 Maintenance - Storage Space Configuration

Neither a node nor the cluster should ever be allowed to run out of storage space.

At a cluster level, ideally, there should always maintain enough space into which to recover data should there be a single node failure. If there is less space remaining that this, the cluster has an “amber” status.

A cluster will always be considered full if there aren’t at least two nodes left with disk space available. At that stage the cluster will no longer accept data writes.

At a node level, the server will fill a node to 97% capacity. This allows space for log files, auditing, attribute changes and database space. A node that is full will no longer be selected for writes.

A rough formula for calculating whether the cluster goes to amber status (in a cluster with equally loaded, equally sized nodes) is:

*If (space_remaining - space_remaining_on_one_ node) < (total_space * 97%) / num_nodes -> “amber”*

Or, where disks are empty:

% space available = n-1/n

where n = number of nodes.

Or, if there are 4 nodes, keep 25% free. If there are 10 nodes, keep 10% free, etc.

MatrixStore server software contains a registration key license that also limits the amount of space available for use. This allows Object Matrix to freely distribute the software for trials.

Concerning adding storage capacity, the system is only tested for adding extra nodes – Object Matrix does not currently support changing the capacity available on a single node.

4.18 Server Tasks

On each node a number of background tasks run to check and maintain the integrity of the data on the cluster. These tasks include:

- Self-healing
- Data replication
- Data verification
- Regulation compliance

Tasks run in the server layer and can be controlled, to some degree, via the administration screens. E.g., some tasks may be paused, or forced to run immediately.

Data Integrity and the Verify Object Task

When an object is stored to the cluster it is mirrored to two nodes. Transfer checksums verify that the correct data was received at each location and disk flushing is completed before the object ID is returned to client application, however there is a chance that there are underlying disk block corruptions (it is a known that RAID cards are not always able to detect nor recover from such situations, Robin Harris on StorageMojo has several interesting articles on this subject from in-the-field surveys). Therefore, when an object is stored it is listed to be verified.

The verify object task will, approx. 24 hours after the object has been written, verify the object to see that the checksum matches the contents. Should a corruption be noticed, then the object will be moved to a corrupt objects folder, and a replacement will be retrieved from the other mirror copy of the data. If both versions are corrupt (which would seem extremely unlikely) then both copies are left untouched.

An interesting question is how often should objects be rechecked for correct integrity? Too often and the risk is that hardware failure may be invoked by constantly running disks, too long and corruptions may go unnoticed before both copies have become corrupt (or the good node has been replaced).

Standard setting is to recheck the objects are correct once per annum.

Self Healing Tasks (Data Regeneration)

If a node containing a duplicate of the data is un-contactable for a set period of time then self-healing / automated data regeneration will be invoked. The period of time here is a critical factor: the longer the period of time, the more the cluster is in risk of a second failure that causes data the to potentially be permanently lost, the shorter the period the more likely undesirable regeneration will start to occur, e.g., if a node was accidentally powered down for a period of time. Typically the timeout period is set to 72 hours on a raided disk solution.

Thus, should a node be offline for more than 72 hours, the other nodes in the cluster will begin to regenerate any data that they share with that node. All nodes do this in unison, thus in a ten node cluster, on average the other 9 nodes will hold 1/9th of the data held on the down node. If the node is fairly large and filled to 80% full - e.g., 40TB in size, then each other node will need to regenerate 40TB * 80% * 1/9th = approx. 3TB. If data is regenerated to a new location at 80MB/s then the regeneration period will be approx. 10 hours. This is astonishingly quick compared to just about every other clustered storage solution out there.

It is possible to stop automated data regeneration (e.g., if you know a node is going to be offline for a period of time). To do this, simply go to the task pane on the Administration tool, select and pause the Regeneration tasks.

4.19 Node Re-attachment (Version 2.4 onwards)

If a node that was regenerated is reattached to the cluster then the data that was regenerated elsewhere in the cluster, and that is on that reattached node, is moved to a "to-delete area". The administrator can empty that to-delete area via the administration console tasks screen.

4.20 Node Decommissioning (Version 2.4 onwards)

A node may be decommissioned because it is seen as going faulty or because it is simply being phased out. Whilst the node could simple be switched off and unplugged, it is less susceptible to risk to decommission the node from the administration console. (e.g., if the node is simply unplugged, then data could be lost if another node were to subsequently fail before regeneration took place). Also, if "Single Instance Vaults" (see below) are being used then it is very important indeed to decommission a node rather than just switching it off.

4.21 Single Instance Vaults (Version 2.4 onwards)

When a file is copied from the client on to the cluster two copies of that file are made at two separate locations, metadata and policy information are added, resulting in two object instances.

From MatrixStore v2.4 onwards the administrator can elect to make a vault a single instance vault, in which case the following will occur:

- If the vault is a replicated vault, both instances will be stored and verified. The object will then be replicated to the remote cluster, where another two instances of the object will be stored. At that point the source cluster knows that one instance of the object can be replaced by a stub. Should the good instance of the data be lost, then the stub will attempt to fetch the object from the remote cluster to recover the data.
- If the vault is not replicated then once the object has been verified as not corrupt, one instance will be replaced by a stub. Should the good node instance be lost then the data will be permanently lost.

This powerful store and forward functionality of MatrixStore allows up to 50% data space to be saved at any one cluster location. A vault may manually be changed from being a dual instance vault to a single instance vault at any time.

Note that the level of data protection with a single instance vault is strongly compromised. Should a node in the cluster become unavailable (e.g., via a local filesystem corruption), then data will become irrecoverable. Therefore data should only ever be kept “Single Instance” if there is another copy of the data available elsewhere, or if the data can afford to be lost.

4.22 Removing MatrixStore Software

An archive solution should store data in an open, and well-tested format; not in a format that is subject to frequent change (as tape formats are) and not one that depends on the support of an individual company.

Since MatrixStore stores data in an open format it is possible to remove the software, reboot the node and then to access the data directly.

To do this:

Switch off all the nodes

One by one, load the desired operating system onto the nodes (e.g., reload Linux/MacOSX from DVD)

Mount the existing filesystems.

Create a script/application to walk the file system and to collect the metadata for the files into a db, spreadsheet, or other format of your choice. Example applications for this are included with MatrixStore / available from Object Matrix, these are documented and are included in the MatrixStore delivery.

Unlike many archive solutions, MatrixStore’s philosophy is that the data is yours, and that you should not be irrevocably tied in to the solution.

4.23 Data Mirroring

Mirroring on a standard device usually involves making two copies of every piece of data stored, one on each volume (or side) of the Raid unit.

This method has several physical advantages for high speed throughput, but also has several disadvantages, including but not limited to:

- The volumes typically have to be of the same size
- If the device is going to be rolled out, then all of the data needs to be moved off of it first, and then all of the indexes pointing to the data need to be changed to point to the new location of the data
- If the device has a problem and needs to be replaced, then data may be lost

MatrixStore mirroring is based upon object mirroring across independent nodes. When an object is stored in one location, it is simultaneously stored on at a second location.

Since mirroring is carried out at an object level nodes/storage volumes can be of any size.

Mirroring across nodes is not optimal in performance compared to using dedicated fast IO throughput hardware, but it is reliable and flexible.

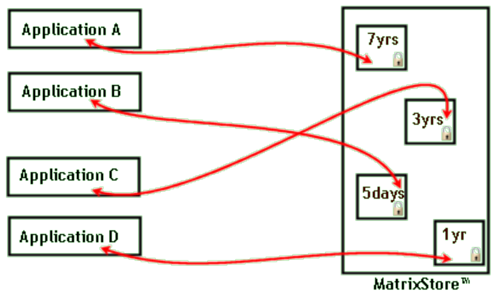
4.24 Regulation Compliance

MatrixStore supports a large number of compliance requirements by ensuring that all data stored is secured from public access, protected against loss, audited, authenticated, available at all times and protected from unauthorised deletion.

MatrixStore helps to support, amongst others:

- Security and Exchange Commissions rule Rule 17a.4 that aims to prevent overwriting, erasure or alteration of records.
- HIPAA privacy ruling for Data Protection, requiring compliant backup methodologies to ensure the security and confidentiality of patient records.

The Sarbanes-Oxley Act of 2002 protecting investors by improving the accuracy and reliability of corporate disclosures. The Act amends mail and wire fraud infractions with harsher punishments and imposes fines and prison sentences of up to 20 years for anyone who knowingly alters or destroys a record or document with the intent to obstruct an investigation.



Individual vaults of data can support different regulations

Some of the ways that MatrixStore helps to support regulations are:

Regulation Compliance Requirement	MatrixStore Solution
Guaranteed retention of data for specific amount of time	Policy stored with data enforces that the data cannot be deleted before its time.
Audit logs and log files	Configurable audit log can track writes, reads, deletes. Log files track other system modifications.
WORM	Data is stored fixed cannot be modified.

Authentication	A 256bit digest calculated when the data is stored acts as a guarantee that the data is bitwise exactly the same when it is read back as when it was originally stored.
Security and Privacy	Full network security is employed to stop replay, sniffing, data modification and other attacks. User access rights can be set to only give access to certain groups of data.
Accessibility	Time to first byte is sub-second, even under heavy load.
Searchability	Built-in database can support many searches per second across 100's of millions of database entries.
Disaster recovery	Covered on two levels: by replication to a separate cluster and data is stored on a single cluster such that 4 disks would need to irrevocably fail before data is at risk.

4.25 Data Replication

It is sometimes desirable for purposes of data protection / disaster recovery (DR) / business continuance to ensure that data is stored in more than one geographic location.

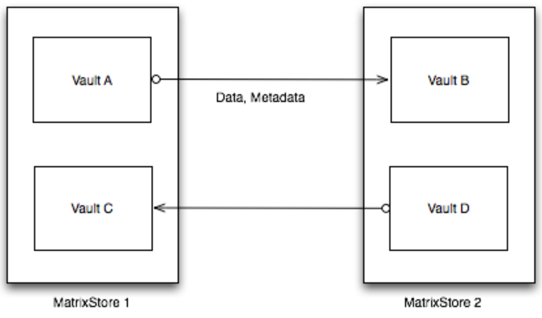
Good replication solutions should provide the following attributes:

- Simple to set-up and use = less human or technical errors
- Not tied to the underlying hardware = flexibility and serviceable in the future
- Neither O/S specific nor data type specific
- Works over standard transport protocols or VPNs
- Secure data transmission options
- Scalable performance
- Easy to select data that needs replicating and data that doesn't
- Ability to replicate deletes or not to
- Facilitate high availability, e.g., by allowing Reads from either the source or target of the replication
- Facilitate fast disaster recovery in the case that one site is lost, that work can immediately continue at the replicated site

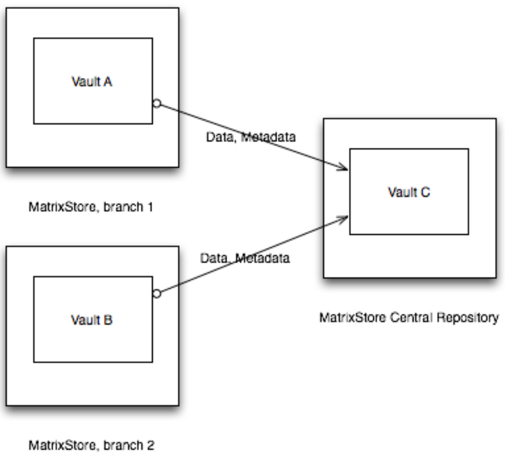
MatrixStore supports and/or facilitates each of the above attributes. Its replication works on an asynchronous model of transmission to the second store. The replication is carried out over a secure (encrypted) TCP/IP connection. Therefore, replication communication can be carried via a public internet provider or via a VPN / private dedicated connection.

Individual vaults within MatrixStore can be selected to be (or not to be) replicated. Thus, not all the data on a MatrixStore cluster needs to be replicated and two clusters do not need to be the same size as one another.

In this first example we show a set up where one vault is being replicated to another MatrixStore, which in turn is replicating another vault back to the first MatrixStore.



A cluster topology might simply contain two clusters, but also, a cluster may receive data from several other clusters so that, e.g., a central data repository could be created that receives data from many branch offices. E.g.,:

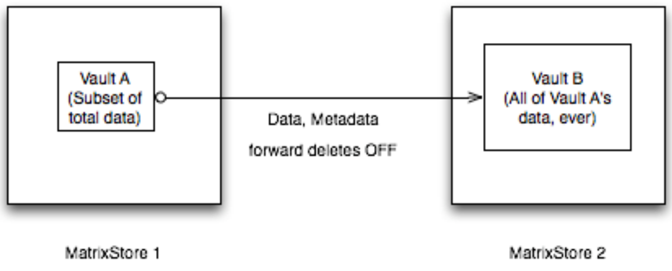


The central repository vault can be attached to for reads.

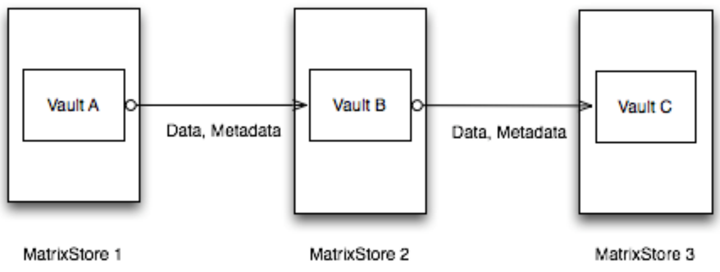
Of course, replication could also be set to replicate to individual vaults at the central repository if separation of the data is required.

Other features are:

- Optionally deletes can be forwarded or not. Thus the Vault at the central repository can become a media library, whilst the original (e.g., branch) location can be kept small.



- A vault can be relayed onwards, such that the data added to a vault can be kept on all MatrixStores.



- If the vault is single instance, two instances will be kept at the source cluster until one instance is effectively forwarded to the second cluster. Thus at no time will the user have a single instance of data. See also Single Instance Vaults, which ensure that only one copy of data per location is kept.

MatrixStore Replication Restrictions

- Two way replication is not supported
- For disaster recovery situations the users of Cluster 1 need to also be kept in Cluster 2

4.26 Management API

The Management API:

- Enables commands that can update / effect the whole cluster
- Ensures that commands reach all the nodes, and that settings are received by nodes that were temporarily offline when the command was issued

The management API is pure Java, and connects to one node in order to initiate instructions that will subsequently be rolled out to the rest of the nodes in the cluster. Commands include adding vaults, changing users, getting audit logs, changing task settings, etc. When a communication is set-up between a client and the server for a management function:

1. Client requests and agrees with the server up to a 256-bit secure channel (see footnote [4])
2. User credentials and 256bit password are transmitted
3. Data pertaining to the operation is transmitted. Data is 256-bit encrypted. (see footnote)
4. Receiving node relays the operation to other nodes in the cluster.

When a node is attached to the cluster it will automatically take the clusters settings from the nodes that are already present.

4.27 MatrixStore API

All communication with MatrixStore goes from the client to the server using MatrixStore's API. This enables us to maintain security, failover, cluster discovery, retries, transport protocol selection, and metadata control.

The MatrixStore API is available in C and Java (the Java implementation being a wrapper for the C implementation). The API is tested upon:

- Windows 2000/XP onwards
- Linux, any distribution with GCC v3.4 or greater and glibc v2.3.3 or greater (e.g., Red Hat 8/9, Suse 8, Fedora, Gentoo) onwards
- MacOSX 10.4.x onwards

The MatrixStore API is not tested upon (but can be tested on demand to work with):

- Unix, any distribution with 'upperbit' version (e.g., Sun Solaris 9, IBM AIX, HP Unix)

Full documentation for programming the API is available upon request from Object Matrix.

4.28 Upgrading Software

Software is upgraded from the MatrixStore Maintenance tool. The tool will inject the software into all the online nodes of the cluster and will restart them to complete the upgrade.

4.29 Cluster Monitoring and SNMP

There are two ways to monitor the health status of the cluster:

- SNMP traps

[4] Note that MatrixStore builds for countries with export restrictions are available that don't include these encryption options

- Via the MatrixStore Administration Application

A client must register the SNMP monitor via the MatrixStore Administration application in order to receive traps.

The MatrixStore Administration application registers the monitor and / or receives and shows health information itself via the Management API.

4.30 Updateable Objects (Version 2.4 onwards)

A Vault can be created or changed to be “updateable”. Unlike other objects in the cluster, objects in an updateable vault can be modified at any time.

This is critical for any application using random access (e.g., a filesystem interface) to write data to the MatrixStore.

Primarily, updateable vaults enable filesystem front ends to be easily implemented.

If a vault is set to be both compliant and updateable, then updates are allowed only for a short amount of time (first 10 minutes).

5 10GigE or not 10GigE?

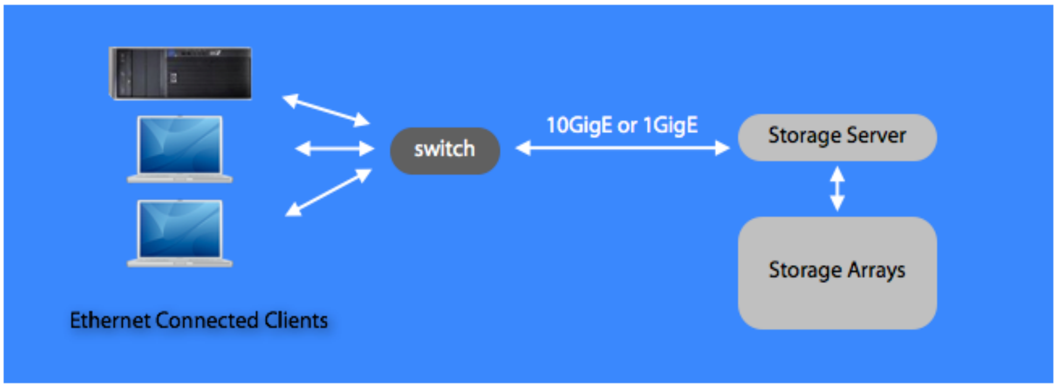
Copied from an Object Matrix Blog article:

A common MatrixStore question is – can we do 10GigE? The answer is both “yes” and “why do you need that?” To understand that response it is good to compare traditional to the MatrixStore way of doing things.

5.1 Traditional Data Flow between Clients and Servers

Traditional connections to a server are of course limited to the bandwidth provided by the “entry point” to the server. 100Mbit/s, 1GigE, 10GigE…

A typical traditional architecture:



Solutions being released this year are still based on that architecture.

The first problem with the traditional architecture is that the solution is limited to the speed of the cable in the solution. The better types of solutions in this class maybe have two or more bonded connections, but ultimately you are talking about only a few clients being able to get full speed access: double the number of clients and you’ll lose half the access speed per client.

Pumping all your data through a single pipeline is akin to everyone flying to New York and then transferring rather than flying direct to your final destination.

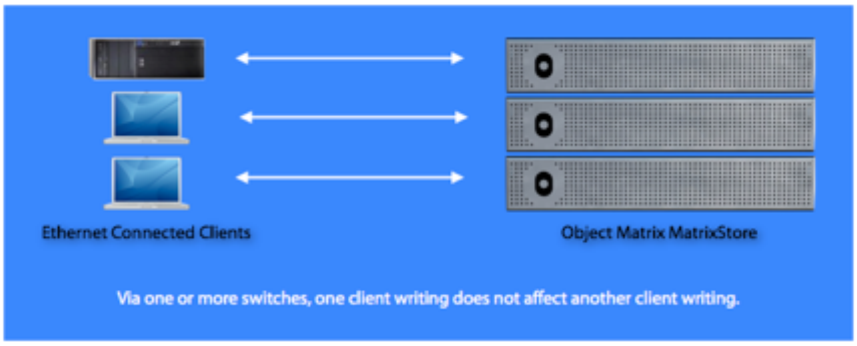
The second bottleneck in a traditional server is also a major issue: how fast can that server process the packets that it receives? If the server is, e.g., utilising software RAID, then the ability of the server to be able to software RAID more than a hundred MB/s is often limited by CPU power. Without the server having some very powerful hardware inside, a single server is not going to be able to get anywhere near 10GbitE/s, however many 10GbitE/s ports it has.

The third problem is that most post-houses and broadcasters simply aren’t set up with 10Ge infrastructure. There might be a few servers with 10Ge but few have anything other than 1Ge clients.

Add to that the obvious problems of having a single point of failure, and the difficulty in being able to expand the location, and you begin to ask if there isn't a better way!

5.2 MatrixStore RAIN (Redundant Array of Independent Nodes) Architecture

With MatrixStore, the server is a RAIN cluster. Typically we sell each MatrixStore node with two 1GbitE connections. But that is limiting - the beauty of the solution is that the clients using the solution are not limited to 1GbitE or 2GbitE at all. Whilst one client is writing to one node another client can be writing to a second node: the result is a combined aggregate bandwidth:

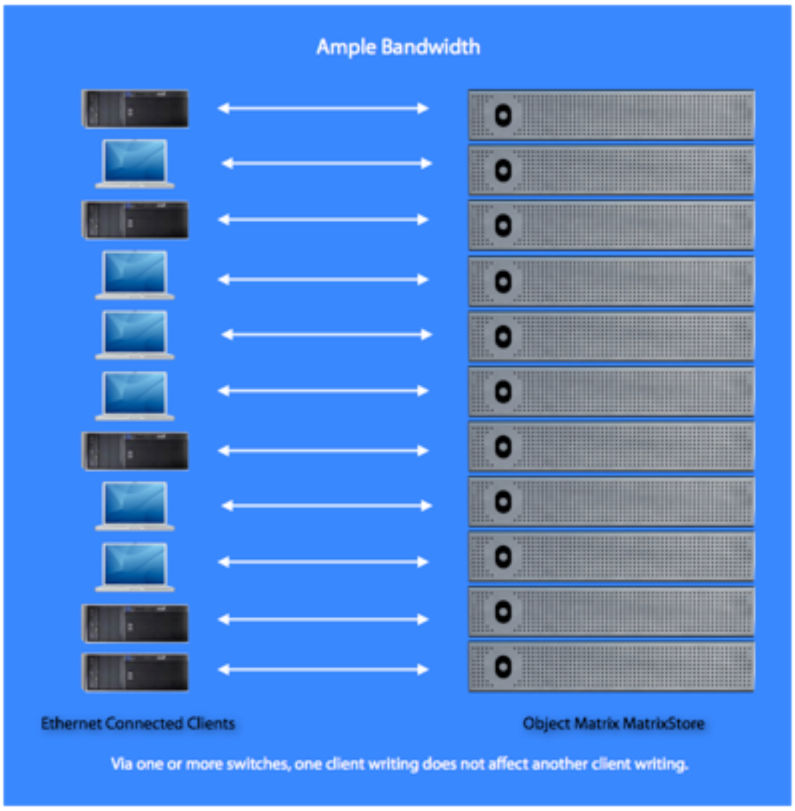

Ethernet Connected Clients
Object Matrix MatrixStore
Via one or more switches, one client writing does not affect another client writing.

Secondly, because the work in the cluster is distributed: there are many CPUs to handle the connections being made including the RAID'ing of the data.

Third, all the clients can be on 1GbitE, but the total bandwidth can be as large as the cluster is.

Even if the data is all coming from the same client could be writing many files at the same time to multiple different locations using different IP connections for all the files being sent.

So, in short, if you need more aggregate bandwidth, simply add more nodes!:


Ample Bandwidth
Ethernet Connected Clients
Object Matrix MatrixStore
Via one or more switches, one client writing does not affect another client writing.

Lastly, if individual files need to be sent with larger bandwidths and if your clients support them, then there is always the option to put individual 10GbitE connections on the nodes in the cluster.