Quantum

Quantum StorNext Scale-Out File Storage

Architecture, Features, and Differentiators

WHITE PAPER

CONTENTS

Introduction

The amount of data being created and managed is increasing at phenomenal rates. According to IDC the “Global Datasphere” in 2018 reached 33 Zettabytes (ZB) and is predicted to reach 175 ZB by 2025. A huge fraction of this data is video, imagery, and similar unstructured content. Contributing to this explosion is the fact that data capture devices—from cellphone cameras to satellite sensors—constantly offer improved resolution, higher fidelity, and greater sampling rates. As a result, not only is data growing, but file sizes and file streams are becoming larger, and higher performance systems are required to store and work with them.

Storing larger and larger files, capturing and serving them faster and faster, has always been a challenge for data storage systems. StorNext® was created over 20 years ago in response to this challenge—specifically capturing, managing and sharing high-resolution digital video and other performance and latency-sensitive data. Driven by the voracious demands of the media & entertainment industry, StorNext has constantly evolved to stay one step ahead. Today StorNext High-Performance File Storage is the de facto standard for storing, serving, and sharing video and video-like content. But due to its performance and flexibility, it’s also suitable for many other data types and workloads.

StorNext is software. A variety of deployment options are available, including traditional Quantum hardware appliances, software-only on customer-supplied compute and storage hardware, "as a service" completely managed by Quantum, or even in the AWS public cloud through the AWS Marketplace. StorNext’s extensive configurability and tunability enable maximizing the potential of any platform it’s deployed on.

After some basic orientation, this white paper will explain the architecture of StorNext Scale-out File Storage, describe many of its features and capabilities, and highlight details and patented technology that make it unique.

StorNext Scale-Out File Storage Architecture – High Level

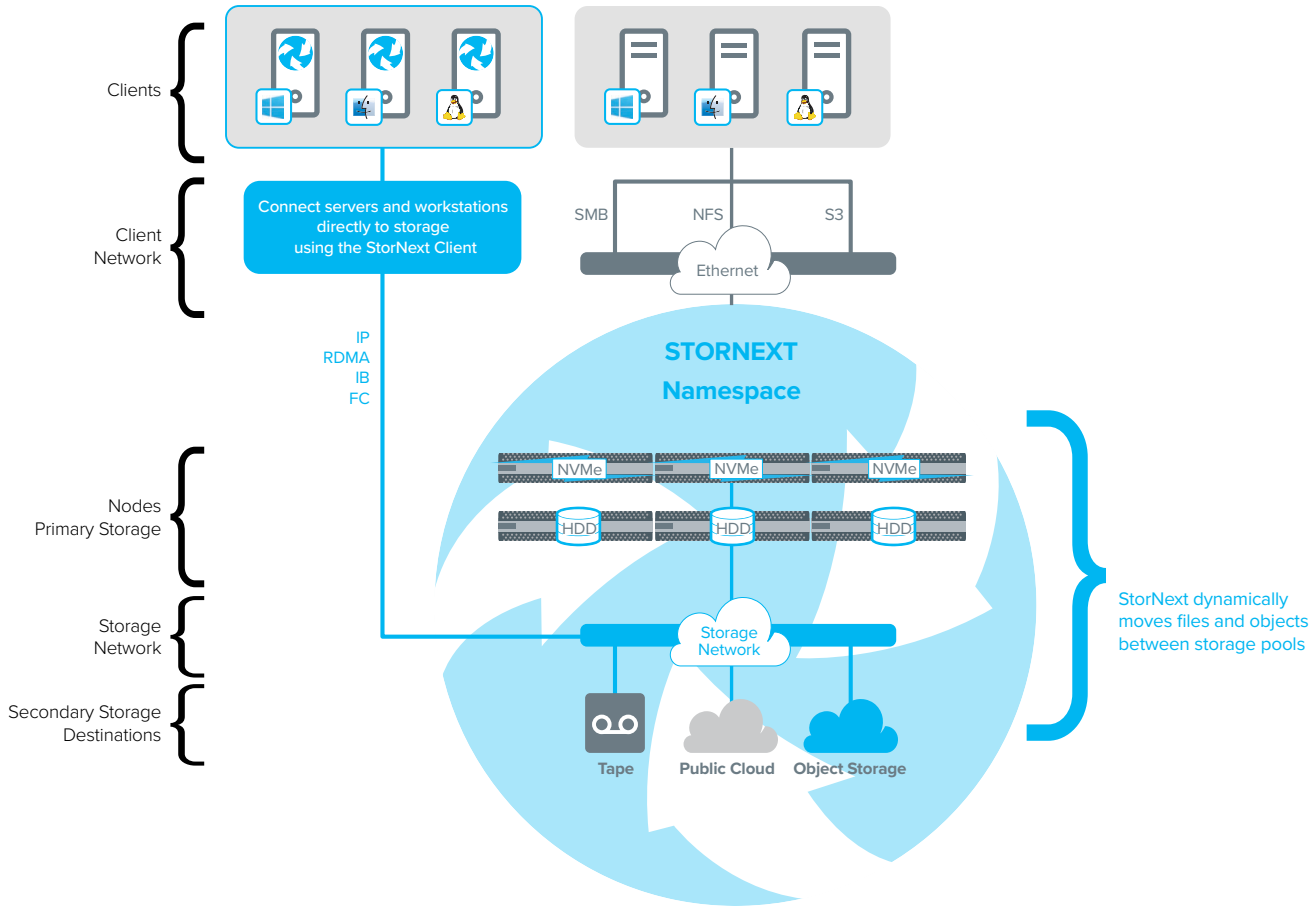


Figure 1 – High-Level StorNext Scale-out File Storage Architecture

At a high level, a StorNext Scale-out File Storage system contains only a few different types of components, as illustrated in Figure 1. At the core are nodes that run StorNext data services and contain compute, storage, or converged combinations of both. In the smallest systems there may be only a single node, but a cluster of nodes is most common, to provide redundancy and scale.

StorNext data services running on the nodes include a policy engine function that manages data copies and storage destinations, including flash, disk, tape, object storage and cloud. Regardless of where data resides in this architecture, it is always visible and accessible to clients in the namespace where it was written. Policies enable automated optimization of storage performance, cost, and protection level.

StorNext clients are servers and workstations that access one or more StorNext file systems. There are several ways for a client to connect, each with specific advantages. For ease of use and ubiquity, it's hard to beat SMB and NFS. Though it's possible to stream content using NAS protocols, they weren't designed for the purpose. For top performance, a system must connect via StorNext client software. Servers and workstations running StorNext client software have the most direct, most highly optimized connection to StorNext file systems. The final connectivity option is

via S3, the de-facto cloud protocol. This allows a portion of the StorNext system to be set aside for use as an object store by any application that speaks S3's language of PUTs and GETs.

Connecting these components are two types of networks. Client networks connect clients to file systems and the storage within. Storage networks connect the nodes together and to external storage destinations. Machines running StorNext client software also connect to the storage network, which is one of the keys to their special capabilities.

The StorNext File System

At the heart of a StorNext Scale-out File Storage system is the StorNext File System, aka SNFS. Much of the power, scalability, and flexibility in a StorNext cluster is due to the capabilities of SNFS. File systems are non-trivial to write, and the stakes are very high. Built well, a file system gets out of the way of the storage, enabling maximum performance while maintaining coherency and data integrity. Built poorly, a file system can hamper performance and even cause data loss. SNFS has been undergoing continuous development, refinement, and improvement for decades, and its safety and performance have been proven in some of the world's most demanding data environments.

A StorNext cluster may host one or multiple StorNext File Systems, depending on customer needs and objectives. Figure 2 illustrates the major components that make up a StorNext File System and how they relate to the underlying storage.

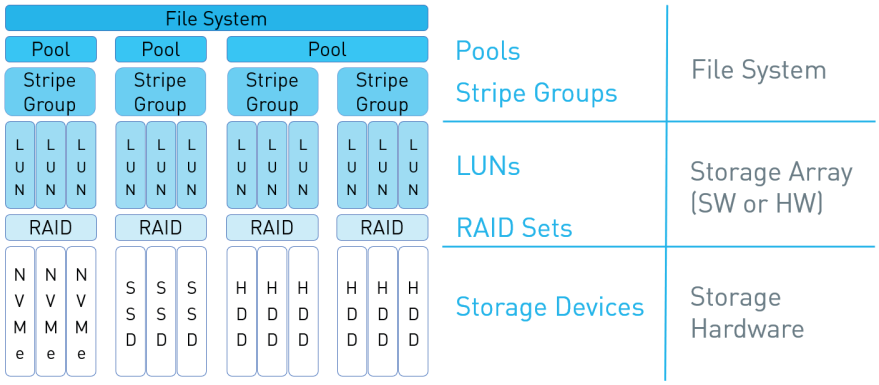


Figure 2 – StorNext File System Architecture

STORAGE DEVICES

SNFS has always been storage device agnostic. It doesn't matter if the storage system contains HDD, SSD, NVMe, or Vulcan memory crystals. If it looks like block storage, SNFS can use it. It is also possible to combine multiple types of storage within the same file system, while maintaining their individual performance characteristics.

RAID SETS AND LUNS

Raw storage devices are almost always aggregated into RAID sets for performance and redundancy. This can be done via the capabilities of a dedicated hardware-based storage controller, or entirely in software in software-defined storage systems.

Once RAID sets are created, Logical Unit Numbers (LUNs) are defined. Depending on the configuration, there may be a 1:1 correspondence between RAID sets and LUNs or a single RAID set may be carved into multiple smaller LUNs. For example, a RAID1 mirror of two drives may be defined as a single LUN, and a large RAID6 set of 12 drives may be carved into four smaller LUNs.

Enterprise NVMe drives present an additional configuration option, using namespaces, analogous to partitions on an HDD. RAID sets and LUNs may be created across a portion of several NVMe drives, as illustrated in Figure 3. This works with NVMe due to its massively parallel accessibility, whereas with HDD partitions it would be a recipe for lousy performance and short device life.

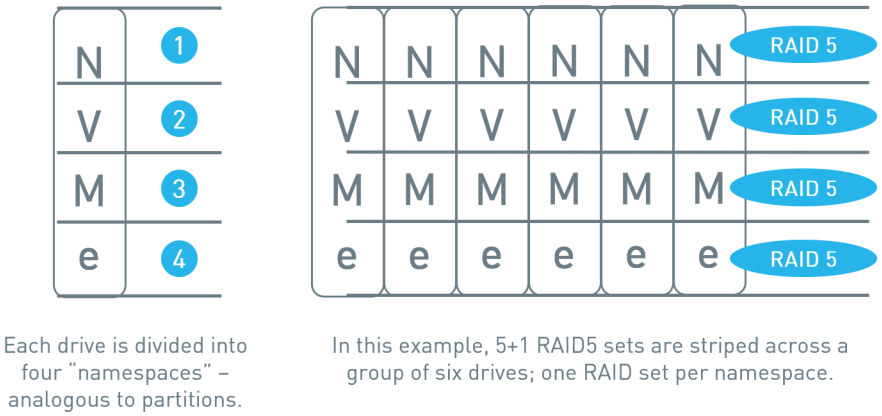


Figure 3 – RAID using NVMe Namespaces

STRIPE GROUPS

Introduction

Storage devices, RAID and LUNs are nothing new in block storage systems. Moving up into the file system structure things get more interesting. Traditional file systems are built on top of a selection of LUNs or other logical devices such as disk partitions. StorNext introduces two layers of abstraction between LUNs and file systems that provide important expanded capabilities; stripe groups and pools.

A stripe group is simply defined as a selected group of identical LUNs. Data is striped across those LUNs, with configurable parameters including:

- **Segment Size:** The amount of data that will be written to one drive in a RAID LUN before writing data to the next drive in that LUN. This is set to match the configuration of the RAID array software or hardware.

- **Data Stripe Breadth:** The amount of data SNFS writes to a LUN before switching to the next LUN within a stripe group.
- **Inode Stripe Width:** If non-zero, causes large files to have their allocations striped across stripe groups in chunks of the specified size.

Many additional parameters are set in the background when using automatic stripe group configuration. Using manual configuration mode enables all parameters to be tuned by the user. Full details are contained in the StorNext Documentation Center.

Stripe Group Types and Use Cases

SNFS uses stripe groups to separate data with different characteristics onto different LUNs. Every StorNext file system has stripe groups that hold three types of information:

- **Metadata** stripe groups hold the file system metadata, consisting of the file name and attributes for every file in the file system. Metadata is very small and accessed randomly.
- **Journal** stripe groups contain the file system journal, the sequential record of changes to the file system metadata. Journal data is a series of small sequential writes and reads.
- **User Data** stripe groups hold the content of files. Access patterns for user data depend on the customer use case, but for video (a common example) are large files accessed sequentially.

In smaller systems with comparatively lower performance requirements, all three types of information may reside on a single stripe group. Typical high-performance configurations combine metadata and journal data, with separate user data stripe groups. In very large and very busy systems, additional performance can be attained by separating metadata, journal, and user data onto individually tuned stripe groups.

There may be multiple user data stripe groups, each tuned for a different type of data or access requirement. For example, video playout requires high-performance streaming with consistent low latency to avoid frame drops. Rendering involves more random access, smaller I/O. With StorNext, you can tune one or more stripe groups for each requirement and still have them be part of the same file system.

Stripe groups also provide a method for scaling a file system in both performance and capacity, because the LUNs in a stripe group do not have to all reside on the same storage system. For increased capacity, a stripe group could span two large-capacity storage arrays, each with many expansion shelves. For scaling performance, a stripe group may span a larger number of lower-capacity storage arrays to take advantage of the aggregate array controller bandwidth. Both techniques are described visually in Figure 4.

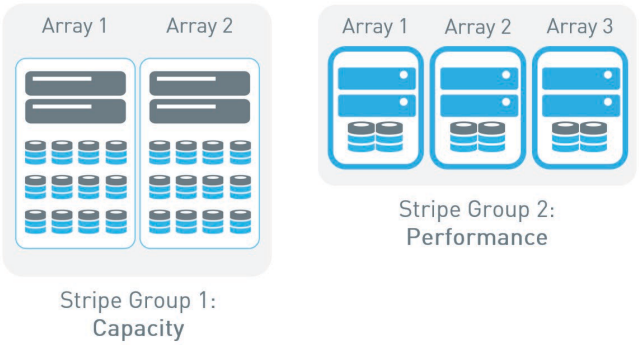


Figure 4 – Scaling Capacity and Performance with Stripe Groups

New stripe groups may be added to an existing file system, and the size of LUNs within a stripe group may be increased when using thin-provisioned storage. The offload function even allows files to be migrated between stripe groups in the background. New storage may be added, and older storage decommissioned while the system remains online.

Finally, stripe groups enable more granular data placement. Within a file system, specific directories or even specific file types (e.g. .jpeg) may be linked to particular stripe groups. These linkages are called affinities. Affinities enable exploiting the unique performance characteristics of different stripe groups while keeping all files logically together in a single file system.

STORAGE POOLS

Introduction and Use Cases

SNFS storage pools are the second layer of abstraction within a StorNext file system. While stripe groups are mandatory, storage pools are optional. Referring again to Figure 2, a storage pool consists of one or more stripe groups, just as a stripe group consists of one or more LUNs. Each storage pool has an associated name, such as “foo” or “bar.”

Storage pools enable transparently moving data between different classes of primary storage, for example NVMe flash and HDD. This movement is triggered manually by an administrator, or automatically by pre-defined policy. “Transparent” means that from the perspective of the clients, the files always remain in their original locations in the StorNext namespace, regardless of which pool they physically reside in.

The value of storage pools is to enable efficiency in use cases where files have different performance needs at different points in their life cycle and make it simple to move content to storage that matches current requirements. For example, non-linear video editing, especially at higher resolutions and frame rates, requires very high bandwidth and low latency. This is a perfect application for NVMe storage. But on a capacity basis ($/TB), NVMe is still significantly more expensive than HDD storage. Storage pools enable leveraging a small amount of fast storage like NVMe when and where needed, bolstering it with more affordable bulk storage for parts of the workflow or data lifecycle that do not require extreme performance.

Another use for storage pools is isolating content from different projects, so that heavy use of one project's files does not affect the performance experience of users working on other projects. Storage pools are also suitable for high-performance ingest scenarios, where data lands on a pool of fast storage at capture time, and is automatically aged off to less expensive, slower storage to make room for new content.

Jobs and Policies

In the context of storage pools, a job is a set of instructions for the system to perform an action on specific files. Jobs may be run immediately or at a specific time in the future. A policy is a request to run a specific job on a regular basis. Policies additionally may be triggered to run only when a pool is above a specified fill level (e.g. 80% full) and may be limited in the amount of work performed per run (e.g. process 1 TB max per run).

The set of files to be acted on may be explicitly specified or based on a specific set of criteria. Criteria may include any combination of file size, age, files or directories, resident pool, and even whether the name matches a custom regular expression (regex) or not. An example policy might be to search the file system for files that are located on the pool named "fast" when its fill level is above 80%, that are older than one week, and move those files to the pool named "slow."

Aside from the system administrator, additional users and groups may be granted User or Administrator access to submit and control jobs and policies. A user can submit jobs, control their own jobs, and view pools and policies. An administrator can perform any action including configuration.

There are several types of jobs:

Job Type	Action
Move	File contents is moved from one pool to another
Estimate	Optionally run before a Move to calculate space needed on the target pool
Promote	Create a new copy of the content on the target pool and retain the old copy
Demote	If Promoted content has not changed, remove the promoted copy, reverting to the source copy. For files that have changed, Move the changed copies to the source pool
Remove	Remove the Promoted content, leaving the source copy intact
Inventory	Count the files and directories in each pool and provide a report
Checksum	Perform a checksum on each file using the specified algorithm and store the checksum in an extended attribute on the file. Stored checksums are externally accessible, enabling custom integration.
Validate	Read and perform a checksum on each file using the specified algorithm, compare against the stored checksum, and report any mismatches

Table 1 – Pools Job Types Summary

Storage pools are an exceptionally flexible and powerful way to control the placement of data within the primary storage environment of a StorNext system and are also compatible with FlexTier secondary storage tiering, described later.

Scaling

Data movement is performed by a system service, and the capacity of pool operations may be easily scaled up by running the service on multiple nodes in the cluster. In the example in Figure 5, the mover service is running on all three nodes of the cluster. Two instances are configured to operate on file system “FS1,”, and the other on file system “FS2.”

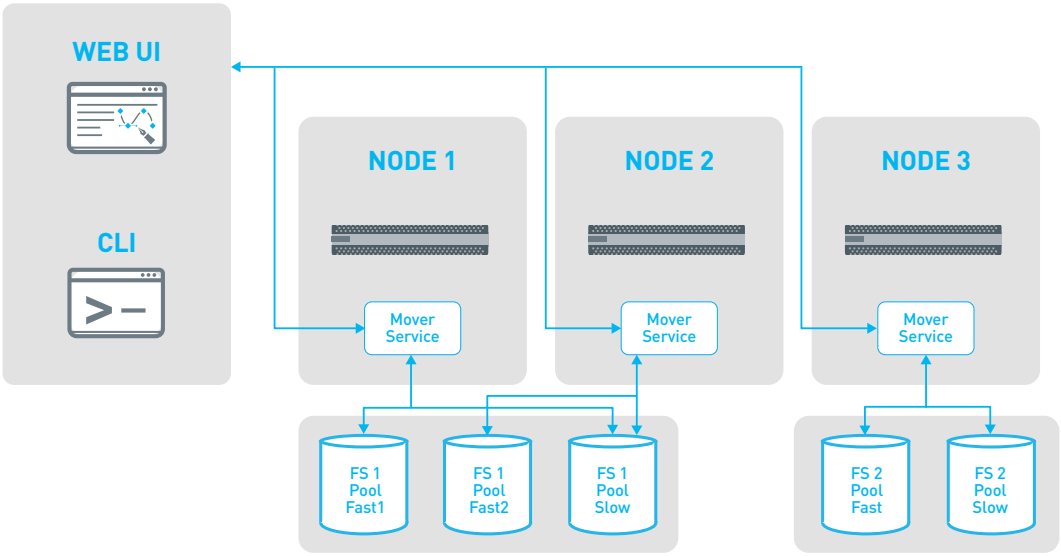


Figure 5 – Scaling Pool Movement Services

FILE SYSTEM

Now that the foundational structure beneath a StorNext File system has been detailed, it’s time to describe some of the key features and characteristics of the file system itself. A StorNext File Storage System cluster may run multiple file systems concurrently, all with unique configurations if necessary.

Metadata

In file systems, metadata (literally “data about the data”) plays a critical role. It consists of file names and locations, but also timestamps, permissions, flags such as read-only and offline indicators, and many other attributes. Shared file systems like StorNext experience the additional challenge of maintaining coherency of metadata, so all users see the same view of the objects in the file system.

All references in this paper to ‘metadata’ refer specifically to file system metadata as described in the previous paragraph. This is different from content metadata – things like artist, track number, video resolution, bitrate, etc. StorNext manages file system metadata, external systems such as media asset managers or other content managers (e.g. CatDV) manage content metadata independent of StorNext.

File system metadata is key to file system performance. If metadata operations are slow, it will limit how fast files can be created or changed. In a StorNext file system, metadata is treated separately from file data. It can reside on separate storage tuned for metadata use, and usually does except in small systems. Metadata communication similarly can happen over a separate network from the data, and always does in high-performance systems. This helps reduce metadata communication latency.

StorNext's efficiencies in metadata communication are covered by many patents, including 8190736, 8554909, 9456041, and 9342568. Metadata communication is discussed further in the StorNext Clients section of this document.

Metadata is commonly cached in memory to avoid the latency in retrieving it from HDD or SSD storage. In file systems with many millions of files, the amount of metadata is correspondingly large. This makes it difficult to cache it in a reasonable amount of RAM. One of the key metadata performance innovations in StorNext is a multi-level metadata cache, including a patented (9176887) compressed level 2 cache. Efficient compression enables the metadata for hundreds of millions of files—or more—to be cached in RAM on the cluster nodes. In all but the largest file count environments metadata read operations are all satisfied from RAM, without having to access slower storage.

Writes of StorNext file system metadata to storage are accelerated using intelligent buffering techniques described in patent 8977590. Common buffering is transparent to the writer, risking data loss if a failure occurs before the buffered data is written to storage. With StorNext's intelligent buffering, the writing client is aware of which metadata is being buffered, and in case of failure or loss of the buffer, the client will re-send metadata, avoiding any possibility of data loss.

The file system journal is similarly tuned for maximum performance and efficiency, using techniques including those covered by patents 9069790 and 9483356.

One of the most powerful innovations in StorNext's metadata handling is the StorNext Metadata Archive (MD Archive). MD Archive is a custom spatial database (patent 10133761) used to store metadata history. Because of the nature of the database and the way the metadata is encoded within it, it is extremely fast to search. Without such a database, some operations, such as finding changes in one region of the file system, would require scanning. For large file systems, scanning can consume large amounts of system resources and take an unacceptable amount of time. StorNext MD Archive enables operations such as replication and defragmentation to proceed rapidly and at low resource cost by replacing scanning with simple queries. Because it also contains the history of changes to the file system, the metadata archive may be used for auditing or forensics to determine "who did what to whom, when." Scanning can never be eliminated entirely, because it's an important way to verify that the file system and MD Archive are in sync, but it's frequency may be greatly reduced. When scanning is necessary, special parallel techniques (patent 14922432) are used to perform it as efficiently and rapidly as possible.

Allocation

Deciding where to store data on the storage—allocation—is another important job of any file system. It's important because it impacts performance. Not just performance of the initial write of data, but read performance, and the performance of the file system over time, because it affects the occurrence of fragmentation. StorNext contains significant intellectual property in this area as well.

SNFS allocates space "optimistically." It assumes that if a write occurs to a file, another write to that file is likely to occur later. In order to keep the file data together and attempt to prevent fragmentation

of both data and free space, more space is allocated than requested. For example, if a 1 MB file is written, 2 MB may be allocated. The amount of extra space initially allocated depends on the file size and is expanded in increasing increments up to a specified limit. Figure 6 below shows an example where the first allocation is 2 MB, the second is 6 MB (2 MB plus a 4 MB increment), and the third is 10 MB (2 MB + 2x the 4 MB increment).

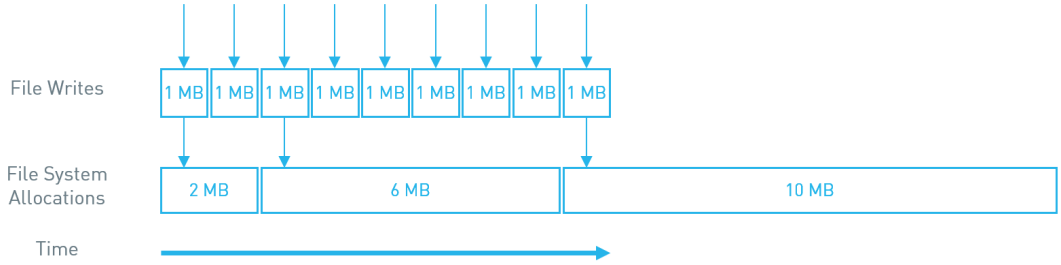


Figure 6 – Optimistic Allocation Example

If needed, the optimistic allocation algorithm may be easily tuned to optimize for specific write patterns. Tools are provided to view allocation statistics to aid in tuning. More details regarding optimistic allocation are contained in the online documentation here.

Allocation Session Reservation (ASR) is a patented (8271760) allocation technique that prevents file system fragmentation and increases performance for use cases which write and read sequences of files in specific directories. This pattern is commonly seen in rich media streaming applications and elsewhere. With ASR enabled (the default), file sequences in a directory are placed on disk in the order in which they are written. ASR keeps these files together even if other applications are writing at the same time in different directories or from different clients.

Without ASR, multiple clients or applications writing file sequences simultaneously can cause storage "checkerboarding", where unrelated data is interleaved. Checkerboarding can have a serious negative effect on performance when files are read. Because these applications write and read multiple files in sequence, a simple "defrag" operation designed to make individual files contiguous has no effect. ASR prevents fragmentation of the entire sequence of related files at the time they are written. In addition to reducing fragmentation of data on the file system, ASR can reduce free space fragmentation since collections of files which are written together are typically removed together.

Figure 7 – Allocation Session Reservation Example

Because the way data flows through every customer's operation is unique, StorNext includes a collection of features and tools to both prevent fragmentation and manage the fragmentation that inevitably occurs in any file system as it ages. A summary of these tools is located here.

Another challenge to performance occurs when applications execute I/O that does not match configured RAID stripe sizes. In some instances, this is unavoidable, as when dealing with "file per frame" video formats like DPX. RAID software calculates parity on full stripes. When a write comes up "short," the RAID software must first read the rest of the existing data in the stripe from storage before calculating the new parity. The delays and additional disk I/O required create significant latency. StorNext uses a unique strategy (patent 8650357) to fill out short writes and prevent the extra reads that would otherwise occur in this situation.

Security Model

There are two predominant security models in modern file systems: UNIX permission bits, and Access Control Lists (ACLs). StorNext supports both models, but when they are used depends on the client OS and the SNFS configuration settings. The selection of security model is made on a per-filesystem basis. Which model is appropriate will depend on security requirements. UNIX permission bits are simpler, but not as flexible as ACLs. Both models allow a single type of permissions to be enforced uniformly across all platforms, providing a consistent experience in heterogeneous environments.

With the ACL security model, permissions are enforced on Windows systems based on ACLs. On Linux and Mac platforms, the security check is based on a combination of ACLs and UNIX permission bits. With the UNIX permission bits model, permissions are enforced on all OS platforms (including Windows), based on UNIX permission bits. For Windows, the UNIX permissions are displayed as a synthesized ACL in Windows Explorer, with one Access Control Entry (ACE) for the owner, one for the group, and one for everyone.

For more details on StorNext security models, including identity mapping, cross-platform permission enforcement, and other topics, refer to the StorNext Security section of the StorNext Documentation Center on Quantum.com.

StorNext Clients & Connectivity

INTRODUCTION

Machines and processes that consume the shared storage services provided by a StorNext cluster are referred to generically as client machines or simply clients. SNFS is a heterogeneous file system, so client machines may run Windows, macOS, or Linux. Client machines connect to the StorNext cluster using some form of client software, either built into the OS or installed separately. Each client connection method has distinct advantages and characteristics as outlined in Table 2. A StorNext environment commonly includes a mix of clients using different connectivity methods according to their needs.

Client Connection via	Performance	Path to Storage	Client Software	Data Network
NAS (NFS/SMB)	Good	NAS Gateway	Built into OS	Standard Ethernet
StorNext Client Software - Proxy Mode	High	StorNext Proxy Gateway	Installed Separately*	Standard Ethernet
StorNext Client Software - Direct Mode	Highest	Direct	Installed Separately*	Fibre-Channel, IB, iSCSI, RDMA Ethernet
S3 Client Software	Good	S3 Gateway	Installed Separately	Standard Ethernet

** macOS includes XSAN, a version of StorNext client software. Windows & Linux require StorNext client software installation*

Table 2 - StorNext Client Summary

The following StorNext architecture diagram provides a more detailed view of the software and hardware components, including the various clients and client connections.

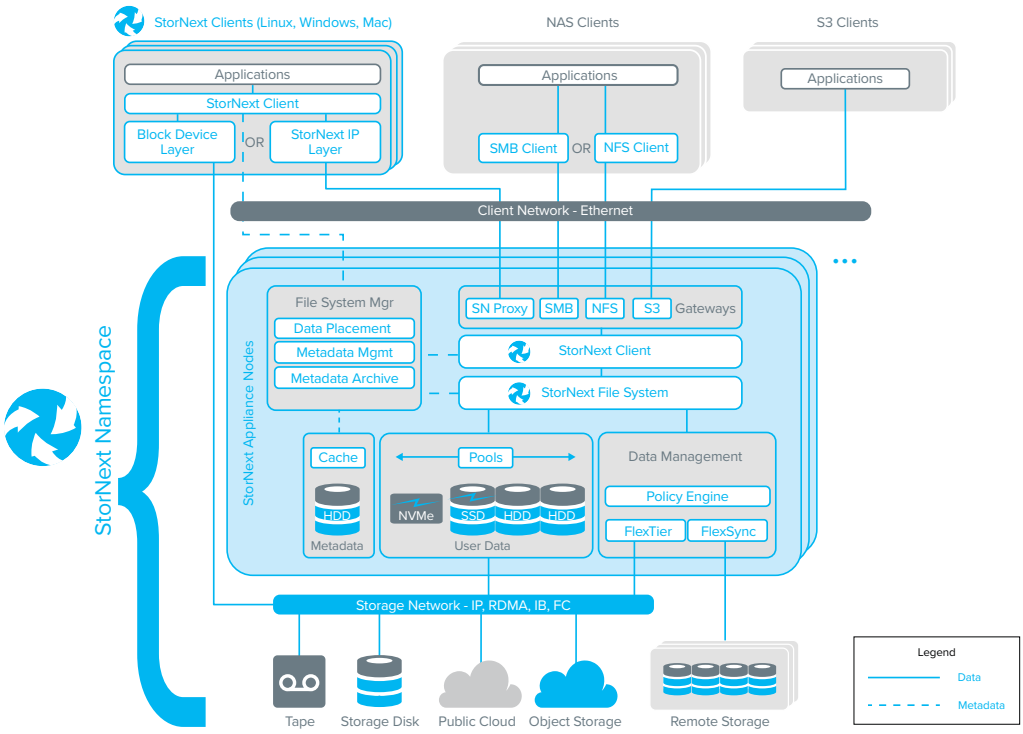


Figure 8 – Detailed StorNext Architecture

NAS

The first client connectivity option is NAS, using standard NFS (v3 & v4) or SMB (v1-v3 including multichannel) protocols. NAS connectivity is simple and requires no code installation, since the ability to connect to NAS shares is included in modern operating systems. It works over inexpensive TCP/IP networks that everyone already has in place, and it's reasonably fast, making it suitable for a wide variety of users and use cases. However, it's not good enough for truly high-performance workloads involving large files or latency-sensitive streaming.

One reason NAS doesn't perform well in these applications is that it's not very efficient in its use of network bandwidth. There is a lot of protocol overhead per unit of data transmitted. Another reason is that NAS client and server software is designed with compromises, so that it can perform well for most users and use cases most of the time. It's optimized toward supporting large numbers of users and relatively small requests.

In a StorNext cluster, one or more nodes may be configured as NAS gateways hosting NAS server services. If multiple nodes are configured, they operate in a redundant and load-balanced scale-out manner using an internal DNS server and virtual IP addresses (VIPs). If a node fails, existing connections are moved to surviving nodes. If a node returns online or a new node is added, connections are automatically rebalanced.

Figure 9 shows an eight-node StorNext cluster where four of the nodes are configured as NAS gateways, with VIPs used for load balancing and failover. Clients always contact the cluster on the cluster VIP (10.1.1.1 here), and the master node distributes the load. If the master node fails, master duty is usurped by a surviving node and connections are rebalanced across the remaining nodes.

More details on the operation of StorNext scale-out NAS clusters are available here in the StorNext Documentation Center.

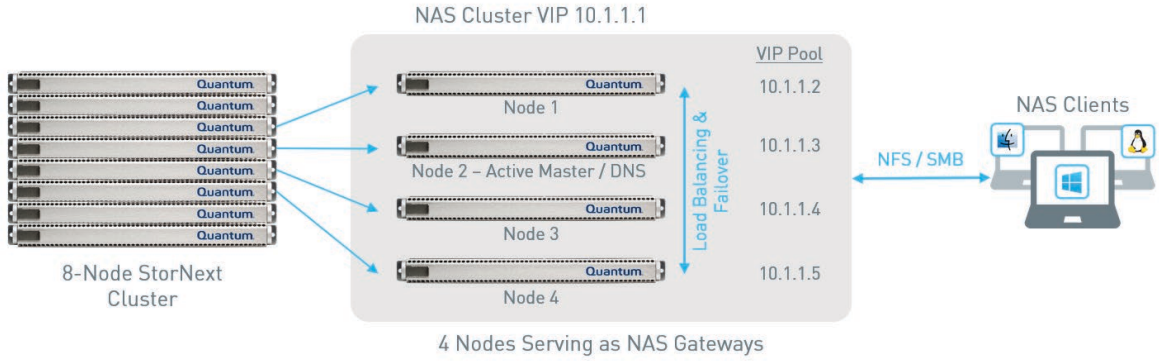


Figure 9 – StorNext NAS Cluster

STORNEXT CLIENT SOFTWARE

The highest performance method for connecting to a StorNext cluster is with StorNext client software. Apple machines may take advantage of the XSAN client built into macOS for this purpose. Windows and Linux machines require a simple code installation.

When using StorNext client software, mounted StorNext file systems appear to the client machine as local file systems, not a remote NAS mount, as shown in Figure 10. The fact that the StorNext File System is remote and shared is transparent to applications.

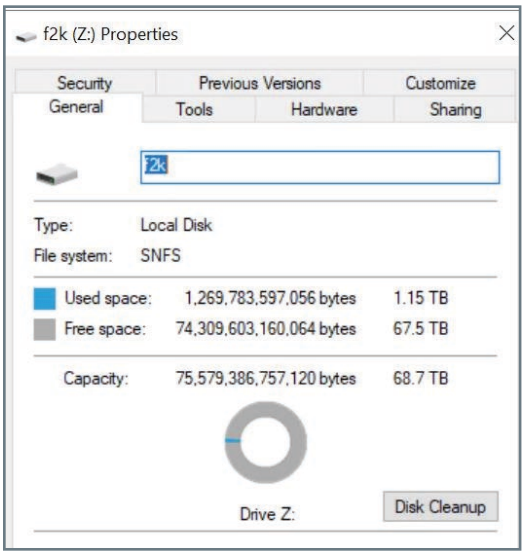


Figure 10 – SNFS Properties Page on a Windows Client

In addition to transparency, StorNext file systems mounted via the StorNext client exhibit much higher performance than NAS mounts, especially for large files and streams. Because StorNext code controls both ends of the connection, a much more efficient transmission scheme can be used, with larger data transfer sizes and lower overhead. The StorNext client and server-side software are designed specifically for the highest performance in streaming applications, an area where traditional NAS falls short.

Connection Mode Details

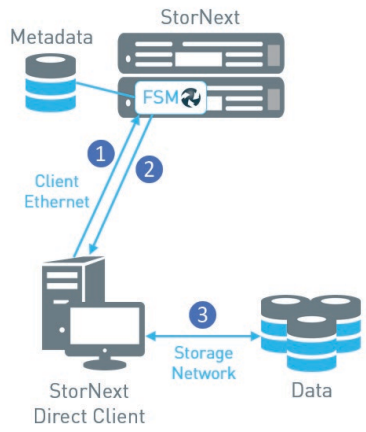
StorNext client software may be run in either direct mode, or proxy mode. The difference has to do with how the client machine is connected to the shared block storage. Clients running in direct mode are known as direct mode clients, or just direct clients. Clients running in proxy mode are referred to as proxy mode clients, or just proxy clients. Figure 8 visually illustrates the details which are covered in the text below.

Direct Mode

Direct mode is the original method of connecting to a SNFS, and is the shortest, fastest path between the client and the block storage. Direct clients connect directly to the shared block storage via the back-end storage network, which may use Fibre-Channel, RDMA, iSCSI or InfiniBand. Read and write activity is straight from the client to the storage, with no "NAS head" or bulky software stack in the way. Aside from the fact that there is a storage network connecting the client to the storage, this is analogous to the way a server writes to local internal storage.

The storage behind a StorNext file system is shared, however, unlike local storage in a server. Shared storage requires a method to ensure that coherency is maintained, so all clients always hold the same consistent view of the file system. With StorNext this role is performed by File System Managers (FSMs). In addition to connecting to the block storage via the storage network, direct mode clients maintain a TCP/IP connection to the FSMs for metadata communication. FSMs are the "traffic cops" that ensure file system coherency.

Figure 11 provides a simplified view of the data and metadata traffic between a direct mode StorNext client, an FSM, and the storage, for both read and write operations. The key aspect to note is that FSMs are not in the data path. Once a client is issued an allocation of blocks by the FSM, the client reads and writes directly to the block storage, only needing to contact the FSM when an additional allocation of space is required.



No.	Read	Write
1	Where's the data for file ABC?	Where can I write?
2	FSM: Block XX to YY	FSM: Block AA to BB
3	Read data direct from storage	Write data direct to storage

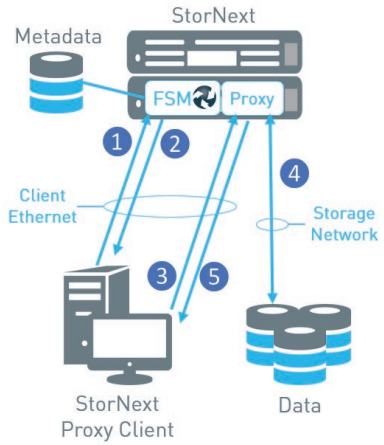
Figure 11 – StorNext Direct Client Communication Flow

The advantage of direct mode is the fastest performance. The downside is cost. Connections to the high-speed storage network cost more than connections to the front-end client network. Yet NAS is too slow for many purposes. Fortunately, there is a third option.

Proxy Mode

Proxy mode, formerly known as DLC, combines faster-than-NAS performance with low-cost networking and unique load-balancing and scalability features. This makes it ideal for cases where some users need very good performance and low latency not available with NAS, but don't need the highest performance level offered by direct mode.

Unlike direct clients, proxy clients are not connected directly to the storage network. Instead, data is brokered through gateways running on the StorNext cluster. The proxy gateways have direct access to the block storage, just like a direct client. When a proxy client needs to perform I/O, it first contacts the FSM over the designated metadata network to obtain the location information on the storage. The proxy client then passes a data request to a proxy gateway, which accesses the storage on behalf of the proxy client, returning data and status as required. This process is illustrated graphically in Figure 12.



No.	Read	Write
1	Where's the data for file ABC?	Where can I write?
2	FSM: Block XX to YY	FSM: Block AA to BB
3	Read blocks XX to XX+n	Write block AA to AA+n
4	Proxy fetches data blocks	Proxy writes data blocks
5	Proxy returns blocks to client	Proxy confirm to client

Figure 12 – StorNext Proxy Client Communication Flow

Having a gateway between the client and the storage sounds just like NAS, so how do proxy clients achieve greater performance? This is a fair question, and there are several elements to the answer.

- The proxy client software stack is very streamlined compared to a typical NAS stack and consumes far less CPU.
- The unique protocol used by proxy clients is much simpler than SMB or NFS.
- Most implementations of SMB are single-threaded per client for metadata requests, limiting performance.
- Even SMB multi-channel cannot split I/O across nodes, so performance is limited to that of a single NAS node.
- Proxy client load-balancing functionality is inherent in the unique communications protocol used. NAS load balancing is DNS-based, resulting in higher latency.
- With StorNext proxy clients a high level of parallelism is possible, even for a single client and single stream, as discussed in the following paragraphs.

Parallelism in the architecture is key to the scalability, load balancing, and resiliency features of proxy mode clients. Large I/Os are sliced up at the client into smaller concurrent I/O requests that are issued across multiple proxy gateway nodes. This increases throughput by using multiple NICs in parallel, even for a single request. To optimize performance on very high-speed (40/50/100 GbE) links multiple connections are made in parallel over each NIC as well, enabling better leverage of multiple CPU cores.

On the proxy gateways, loads are balanced based on queue depth. If the I/O queue to a specific gateway node begins to grow and is not fulfilled quickly enough, the proxy client shifts subsequent I/O to different gateway nodes. This mechanism provides for real-time load balancing based on gateway node latency and throughput and eliminates hot spots. Nodes that process requests faster will receive more client requests. When all queues are empty, individual proxy mode clients will pick a single gateway node at random to service requests.

As additional gateway nodes are added, proxy clients are automatically informed and can start using them for I/O requests. If a gateway node becomes unavailable, proxy clients will continue to use the remaining gateways. I/O that was in progress on the failed gateway is re-issued automatically.

In more complex environments where certain clients or groups of clients require an assured level of performance, a level of manual control may be desirable. To facilitate this, clients may be configured to use a specific subset of gateway nodes or NICs. This allows segmentation of resources without sacrificing load balancing and resiliency.

S3

The third option for connecting to a StorNext file system is via S3, using any S3-aware application or utility. This option enables StorNext to act as a simple object store, through validated and documented integration with MinIO, the popular open source high performance object storage application. This shouldn't be confused with the ability of the StorNext policy engine to write data out to an object store, a capability discussed later in this document.

Object data resides in a dedicated subdirectory of the file system, for example `/minio`. Beneath the object root, buckets are represented by directories, and objects correspond to files. This makes it easy for NAS and StorNext clients to browse the object hierarchy and access objects as files if needed.

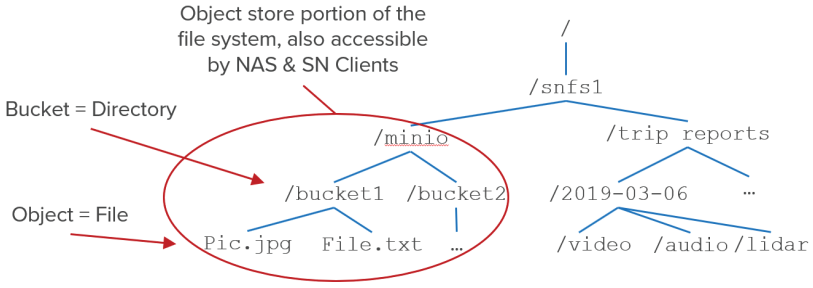


Figure 13 – Object Store Structure: File System View

Access to object stores is via designated cluster nodes that serve as S3 gateways, providing a highly available connection to outside applications. Combining S3 access with StorNext FlexTier data management policies enables offloading objects from disk to tape for long-term archival storage that's less expensive than public cloud cold storage.

Data Services

In addition to the high-speed file system, StorNext contains a robust selection of integrated data services. Some are used for day-to-day storage resource management, others for functions such as data protection, data lifecycle management, or import / export to and from other systems.

Although StorNext data services may be controlled manually by an administrator or programmatically by an outside application, usually they are driven by StorNext's integrated policy and scheduling capability. For example, a storage policy may specify that five minutes after a file in directory /foo is changed, a copy of it shall be made to a local object store and another copy to public cloud, with 10 versions kept as further changes occur. After 12 months the local copy is expired.

The simple policy just described accomplishes comprehensive on- and off-site data protection and helps control storage cost by moving files to a cloud archive as they age.

Table 3 contains a summary of StorNext data services, which are described in more detail below.

Data Service	Description	Purpose
Data Migration	Move files between stripe groups	Resource Management
Quotas	Control sharing of file system capacity	Resource Management
QoS/Bandwidth Management	Control sharing of file system performance	Resource Management
FlexTier	Transparently extend StorNext file systems to secondary storage	Resource Management Data Protection Data Distribution
FlexSync	Directory replication / synchronization	Data Protection Data Distribution
Import/Export	Import files from, or export files to an external system (Tape, Object Storage, other archive SW)	Data Interchange Data Migration

Table 3 - Summary of StorNext Data Services

DATA MIGRATION

While a StorNext Scale-out File Storage system can grow and change to meet evolving requirements, the hardware components within are subject to technical obsolescence. This is particularly true of storage devices. It's common to replace storage arrays every two to three years as higher-performance and more cost-effective technology becomes available.

With SNFS, additional storage configured as one or more new stripe groups may be added using file system expansion. After adding the new stripe groups, there are two options for migrating data. Files from a source stripe group may be redirected to a specific target stripe group, or files from one or more source stripe groups may be spread among all remaining stripe groups in a single operation.

In both cases, the migration process first sets the allocation status (alloc) of the source stripe groups to false. This prevents new file extents from being created, and existing extents from being expanded. If the source stripe groups were simply marked read only, applications that write in place would fail. Using the alloc flag avoids this.

Once alloc = false on the source stripe groups, file content is migrated in the background while the file system is live. If a source file is open, the client is notified to briefly pause I/O while the in-use extent is moved to the new stripe group. The client is then notified to refresh and resume I/O.

When all files have been migrated, the source stripe group may be re-deployed for another use within the system or may be decommissioned and removed.

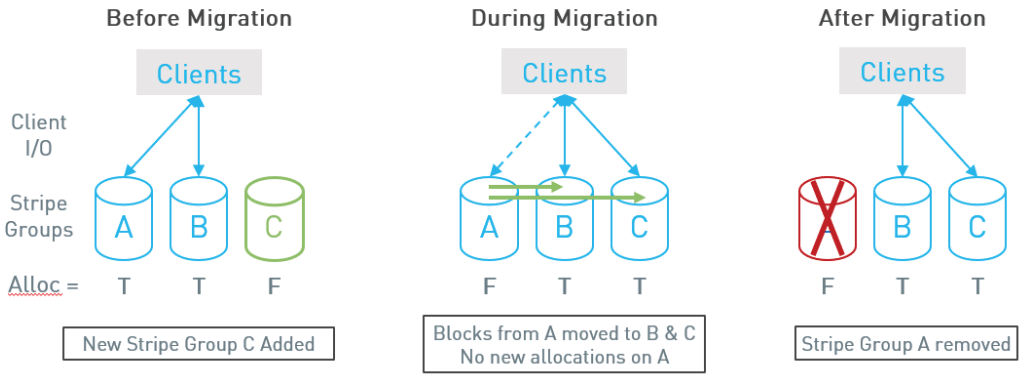


Figure 15 – Online Stripe Group Migration

QUOTAS

StorNext systems are often shared by many users in multiple departments, each with different needs. In a shared environment it becomes important to manage system resources to avoid conflicts. With a shared storage system there are two commodities to allocate: capacity and performance. Quotas are the method used to allocate capacity. Two classes of storage resources may have quotas assigned—primary storage and secondary storage. Quota compliance may be monitored via on-demand or scheduled administrative reports.

Primary Storage Quotas

Primary storage quotas are used to manage capacity on primary storage, such as NVMe Flash, SSD, or HDD. There are three types of file system quotas: user quotas, group quotas, and directory quotas. User and group quotas limit the capacity that can be allocated by a user or group across the file system in total. Directory quotas enable limiting the capacity allocated to a specific directory and its subdirectories, or limiting the number of files, which is sometimes useful.

Each quota may have two limits associated with it, a soft limit, and a hard limit. The hard limit is the absolute limit which space usage should not exceed. Any time the total allocated space is at or over the hard limit, all further write requests by the offending user or group, or within the offending directory, will be denied.

The soft limit is a lesser limit. When the soft limit is exceeded, a configurable grace period timer starts, and a warning is issued. If the soft limit has been exceeded for longer than the grace period, it becomes a hard limit and further writes are denied.

To provide maximum flexibility, it is possible to specify just a hard limit without a soft limit and grace period, or only a soft limit, with or without a grace period. This enables the administrator to create a range of enforcement scenarios, from permissive to strict.

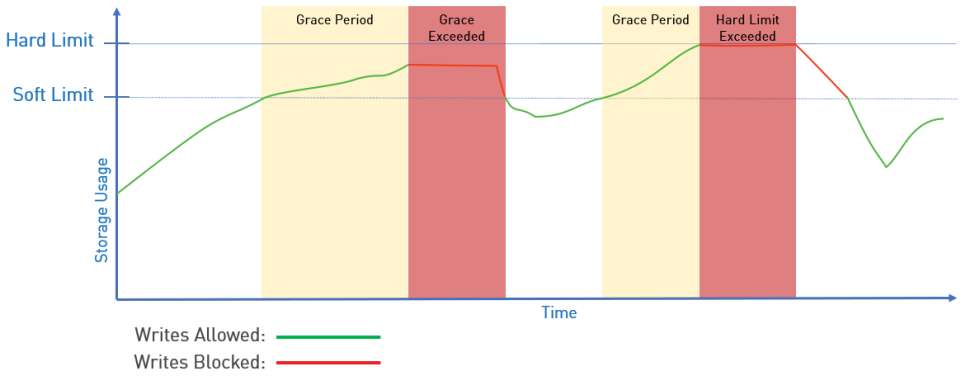


Figure 16 – Storage Quota Limit Behavior

Secondary Storage Quotas

StorNext has a mature and robust set of capabilities for managing secondary storage, as described in the FlexTier section below. Like primary storage quotas, secondary storage quotas provide a mechanism for an administrator to control capacity allocation of secondary storage targets such as cloud, object stores, and tape. Also, like primary storage quotas, secondary storage quotas utilize the concepts of hard limits, soft limits, and grace periods, and monitored entities include users and groups. But instead of simple directory quotas, secondary storage quotas use the concept of projects. A project is a list of one or more directories defined by the administrator, along with their subdirectories. This enables tracking secondary storage usage at different levels of granularity.

Secondary storage quotas may also be applied to specific secondary storage media types (e.g. AWS) or media (e.g. a specific AWS bucket). This can be useful to help understand and control usage of storage billed ‘as a service’.

QUALITY OF SERVICE / BANDWIDTH MANAGEMENT (QBM)

Allocating the throughput of a StorNext system is the function of the Quality of Service / Bandwidth Management capability. There are many reasons one may need to control the performance allocated to clients. Some may require a specific level of storage bandwidth to perform their everyday work. There may be special events that require priority over all other users for a defined time, such as a press briefing or an executive screening. A system with many users browsing content may need a way to protect content creators from being impacted by spikes in viewer demand. StorNext QBM can handle these situations and more.

QBM is configured on a stripe group and client basis. The total bandwidth capacity of each stripe group is defined, and clients are assigned bandwidth based on the class they are part of. Four classes are defined, each with different behaviors. Bandwidth assignment is dynamic, changing continuously based on how the storage is being used. The overall goal is for clients to be able to use all available storage bandwidth (i.e. not restrict it if there is no need to do so) but ensure that higher-priority clients always get the bandwidth they need.

Classes are the QBM configuration building blocks. In addition to class of service, each client may be configured to have a minimum and maximum bandwidth allocation. The combination of total bandwidth available, class, and bandwidth requested determine what bandwidth each client will be allocated.

Class	Priority	Description
First Come	First	Clients receive at least the minimum bandwidth requested or none and will retain their bandwidth as new clients are added. If a new First Come client is rejected due to a shortage of available bandwidth, the client is assigned to the Fair Share class.
Fair Share	Second	Clients share class bandwidth in proportion to their configuration. As clients are added, bandwidth available per-client decreases.
Low Share	Third	These clients share bandwidth not consumed by higher-priority classes. Good for background tasks that can work as resources are available.
Mover	Fourth	A special class used to throttle FlexTier data mover clients so background movement of data to secondary storage does not impact other clients.

Table 4 - QBM Classes of Service

FLEXTIER

StorNext FlexTier is the general name for the subsystem that enables extending a StorNext file system to secondary storage. A SNFS that uses FlexTier features is referred to as a managed file system. “Extending SNFS to secondary storage” means that a system of policies is used to move file data between primary storage and secondary storage destinations as needed, with the result that the primary storage pool appears to be much larger than it really is. The data movement activity happens behind the scenes, invisible (on the surface) to users and applications. Files always appear in the file system where they were written, regardless of where the data blocks really live or whether multiple copies exist. Managing a file system this way saves money, because older data is transparently evicted onto less expensive storage, reducing the amount of expensive primary storage capacity needed. This is the basic value premise of any Hierarchical Storage Management (HSM) system.

But FlexTier is more than a simple HSM function. FlexTier’s ability to create copies and retain file versions makes a StorNext system completely self-protecting, with no need for additional backup software. Files are protected as they are created and as they change.

Secondary Storage Destinations

The general characteristic of secondary storage is that it costs less per TB than primary storage, and usually has lower performance as well. StorNext secondary storage options include Quantum Scalar tape libraries and Quantum ActiveScale™ object stores but are not limited to Quantum hardware.

Many third-party storage devices and cloud services are supported. For the current list, refer to the StorNext Compatibility Guide. Keep in mind that new devices and cloud services are often added by customer request. To request certification of a new secondary storage device or service, contact Quantum.

FlexTier supports the following types of secondary storage:

Technology	Cost	Performance	Description
Disk	$$$	Fastest	NAS or block storage is used as "storage disk" (SDISK) Often repurposed previous-generation primary storage hardware
Object	$$	Faster	On-premises object stores
Cloud	$$	Fast to Very Slow	Off-premises public cloud storage providers, including hot and cold storage tiers
Tape	$	Fast Streaming First Byte Delay	Automated tape libraries using LTO and enterprise drives

Table 5 - FlexTier Supported Secondary Storage Types

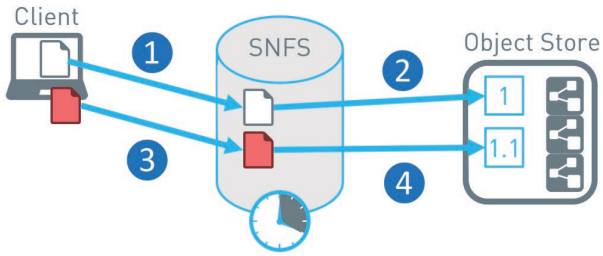
Basic Lifecycle Flow

In a managed file system, file data moves based on administrator-defined policies. This subsection will describe a basic—slightly oversimplified—lifecycle flow, followed in the next section by information on some of the available policy options. Remember that users and applications view the StorNext file system as a single namespace they can use to store and retrieve files like any other file system, and FlexTier activity is not normally visible. Sometimes it is useful or necessary to enable users and applications to see "behind the curtain," and tools such as the Offline File Manager are provided to expose the details of FlexTier operations.

Lifecycle policies are applied at the directory level within the file system. A single policy may apply to multiple directories, and it is not a requirement that every directory have a policy associated with it.

The basic lifecycle flow begins when a file is created or copied into SNFS. The state of the file is monitored, and when it has remained unchanged for a few minutes, one or more copies of the file are stored to secondary storage destinations as directed by policy. If two copies are defined, a total of three instances of the file now exist—the original remaining on primary storage, and the two copies on secondary storage. The file is now protected against catastrophic failure of the primary storage system since other copies exist elsewhere. It's "backed up." If at least one of the two copies is off site (e.g. in public cloud storage), the file is protected against a site disaster as well.

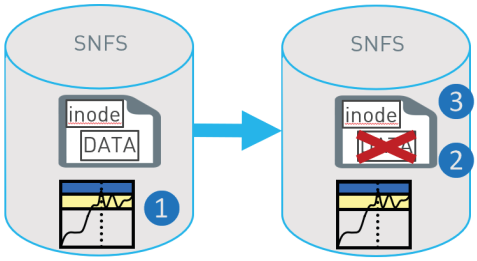
If, after the file is stored, it is later accessed and changed, copies of the changed file will also be stored to secondary storage after a few minutes of quiescence. This creates a version of the file. While copies enable recovering from catastrophic hardware failures, versions enable recovering from more common occurrences, such as accidental modifications. A "trashcan" function guards against accidental deletions.



No.	Action
1	File stored
2	Inactivity timer expires, copy created
3	File changed
4	Timer expires, version created

Figure 17 – Store, Copy, Version Process

Things get more interesting as the primary storage begins to fill. When a high watermark is reached, truncation comes into play. Truncation is the process of freeing space on primary storage by removing the data blocks associated with some files that have been previously stored to secondary storage targets. The inode for the file is retained on primary storage, so the file listing still appears to clients as usual, but the data blocks for truncated files only reside on secondary storage. Truncation runs until primary storage capacity consumption is reduced below a low watermark.

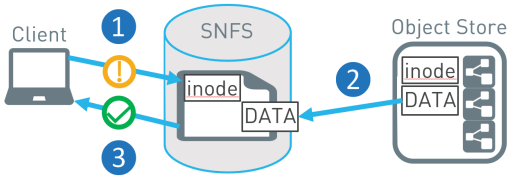


No.	Action
1	Capacity high watermark reached
2	Data blocks for older files removed until low watermark is reached
3	Inodes remain, files visible to clients

Figure 18 – Truncation Process

Truncation ensures that the most recently used files remain on primary storage, while less recently used files reside only on less costly secondary storage. The flip side of truncation is retrieval.

When an attempt is made to access a truncated file, FlexTier charges into action to retrieve the file's data blocks from secondary storage and write them back to primary storage before making the file available to the requestor. The retrieved file then remains on primary storage until it becomes a candidate for truncation again.



No.	Action
1	Client accesses truncated file – access momentarily blocked
2	File data blocks copied back to disk from secondary storage
3	Client granted access to file

Figure 19 – Retrieval of Truncated File

Lifecycle Options

As you read the previous section, your mind may have filled with questions. “Can I tune that option?” “Can I change that behavior?” “What if I don’t want that function, can I turn it off?” As a result of the customer-focused way StorNext is developed, the answer to these questions is almost always “yes.” Nearly everything about StorNext’s behavior can be customized to suit virtually any need. Table 7 summarizes some of the more important options that apply to the lifecycle flow described, but it is in no way complete. For comprehensive details, consult the online StorNext Documentation Center.

Function	Policy Options
Store	Run stores automatically, on user-defined schedule, or on-demand Exclude files from being stored based on saved path & filename patterns Specify file age before copies are stored Delay store until a minimum amount of data is eligible, or a maximum time has elapsed Generate checksums when files are stored
Copy	Number of copies (1-4) Secondary storage destination for each copy Copy expiration (may differ for each copy)
Version	Number of file versions to retain Length of time to keep inactive file versions before removal
Truncate	Minimum age before a file will be truncated Truncate files immediately after they are stored “Pin” specific files to primary storage so they are never truncated Exclude files from being truncated based on saved path & filename patterns Leave stub files (of configurable size) in place on primary storage in addition to inode
Retrieve	Retrieve order - when multiple copies exist (e.g. local object store first, cloud second) Validate checksums when files are retrieved Retrieve files manually (useful for pre-staging files to be needed soon)

Table 7 - Selected Lifecycle Policy Options

Vaulting

Paraphrasing computer scientist Andrew S. Tanenbaum paraphrasing Dr. Warren Jackson, Director, University of Toronto Computing Services (circa 1985), “Never underestimate the bandwidth of a truck full of tapes.” Though less popular than it once was, the most cost-effective way to move a large volume of data in a short period of time is still tape. Vaulting functionality in FlexTier enables moving all tapes holding a specific copy number (e.g. Copy 2) of files out of an automated tape library.

Vaulting provides a way to quickly move a copy of even large archives safely off-site. Many options besides copy number are available to customize what gets vaulted and when and configure notifications and reports to easily keep track of vaulting activity. Vaulting is different from exporting (discussed later) in that FlexTier retains awareness and ownership of vaulted media. If a local version of a file is damaged or otherwise unrecoverable, for example, FlexTier will notify the administrator to return the vaulted copy to the system to retrieve the file.

If security is a concern, tapes may be encrypted, and WORM media may even be used to ensure that data cannot be tampered with while out of the system.

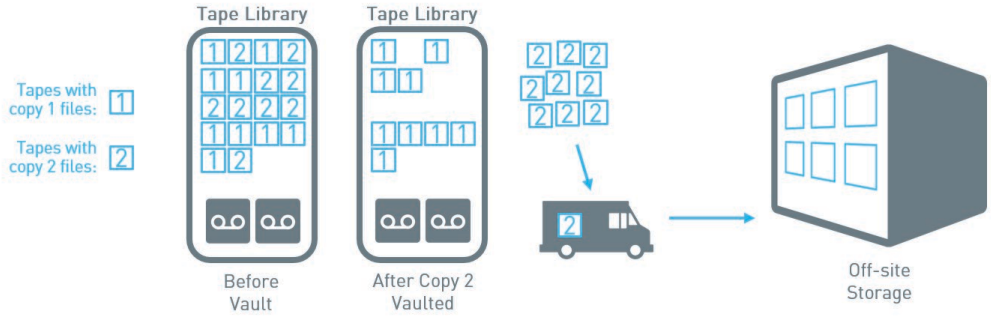


Figure 20 – Vault Process

Vaulting may also be used to move tapes containing rarely used data out of an automated tape library, freeing slots for more recent data. In this scenario tapes are moved to a shelf, reducing the size of the tape library required.

The downside to moving tapes out of the library simply to conserve slots is that manually handling tapes incurs the risk of damage from dropping, spills, dirt, and other dangers. There is a better option. StorNext's vaulting capability is fully integrated with a feature in Quantum's Scalar libraries known as Active Vault (AV). AV enables in-library vaulting, in which vaulted tapes are moved to a separate library partition not accessible to the application. AV uses unlicensed slots, which are less expensive than licensed, application-usable slots.

StorNext Active Vault policies may be configured to automatically move candidate tapes from a StorNext partition into an AV partition based on partition fill-level watermarks. Many parameters are available to select candidates, including the media fill or free percentage and the time since the media was last accessed. When StorNext requires access to tapes residing in the active vault, an administrator is alerted, and may return the requested tapes to the StorNext partition with a simple GUI action, without leaving their chair.

Because the tapes in an AV partition never leave the library, they are protected from physical damage. Because they are stored in slots not accessible to the application (StorNext in this case), they are also "air-gapped," and thus protected from ransomware. For more on using Active Vault to protect data from ransomware, refer to this Tech Brief available on Quantum.com.

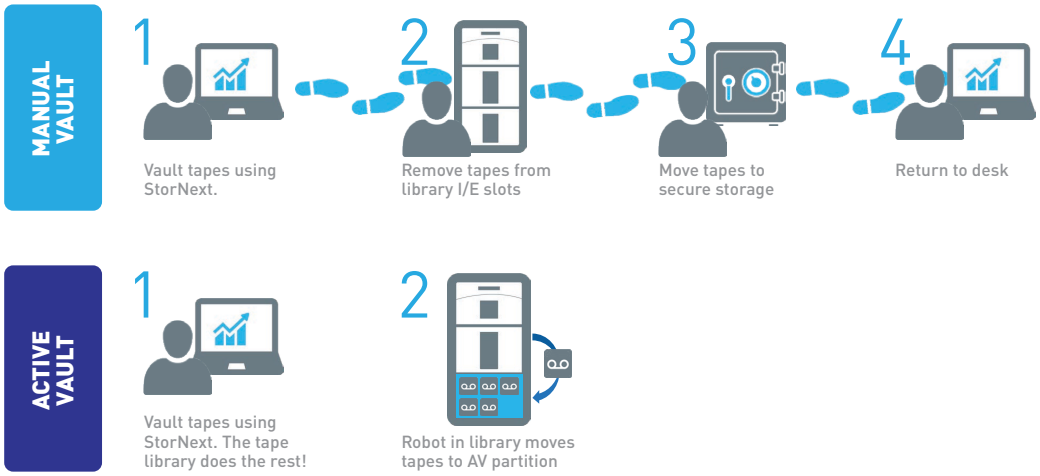


Figure 21 – Active Vault vs. Manual Vault

Offline File Manager

It has been mentioned several times that FlexTier activity is not normally visible to StorNext users and applications, and this is often preferred. Invisibility can be a double-edged sword, however, particularly when cloud storage or tape is involved. Depending on file size and the storage tier used, it can be a few minutes or a few hours after someone clicks to open a file before it returns to disk and is made available. If the user isn't aware that the file resides on slower media, they will be confused and frustrated. In these cases, giving end users visibility into, and limited control over FlexTier operations is vital. Offline File Manager (OFM) is the tool used to provide user visibility and control.

OFM is a client-based tool available for machines running Windows and macOS. It functions as an extension to the existing file browser—Windows Explorer or Mac Finder. Icon overlays provide a quick visual indication of whether a file is still on primary storage or truncated, and if truncated the type of secondary storage housing it. A context menu enables the user to store, truncate, and retrieve files on demand, without administrator involvement. OFM also conveniently works with primary storage tiering to enable clients (including NAS clients) to directly view the pool in which a file resides, and move files between pools on demand.

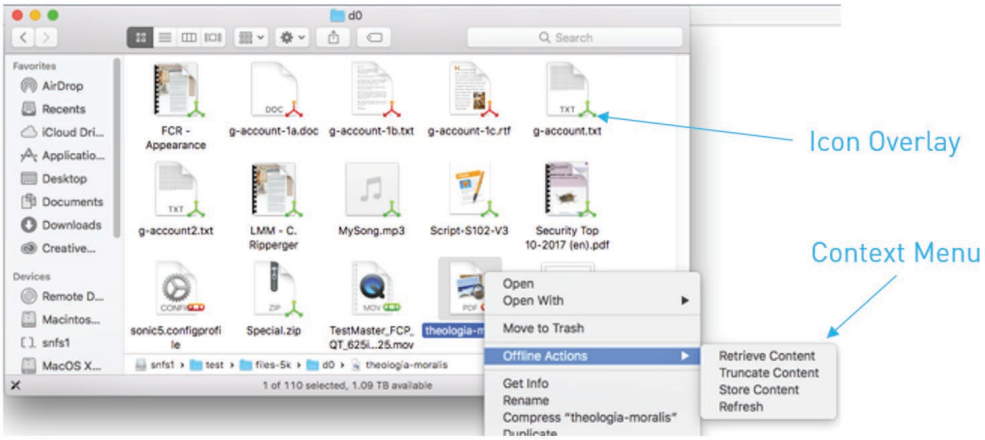


Figure 22 – FlexTier Offline File Manager File Browser Integration (Mac)

Symbol	Description	Symbol	Description
	File is on primary storage File has copies on multiple media		File is not on primary storage (truncated) File has copies on multiple media
	File is on primary storage File copy resides in an object store		File is not on primary storage (truncated) File copy resides in an object store
	File is on primary storage File copy resides on tape		File is not on primary storage (truncated) File copy resides on tape
	File is on primary storage File copy resides on SDisk		File is not on primary storage (truncated) File copy resides on SDisk
	File is on primary storage		File is not on primary storage (truncated)

Table 8 - FlexTier Offline File Manager Icon Overlays

In addition to the file browser integration, a client-side CLI is provided. This enables actions to be scripted. For example, if a user is regularly switching between two large projects, they can write a script to store and truncate all files in the Project A directory and bring all of Project B's data back to primary storage from the cloud archive.

FLEXSYNC

Built to protect data that is managed by StorNext, FlexSync (US patent 10896156) is a simple, fast, and highly efficient tool for creating local or remote replicas of file system data and metadata. It is the only replication & synchronization tool that protects all StorNext-specific metadata, such as Windows ACLs and named streams, providing an absolute 1-for-1 copy.

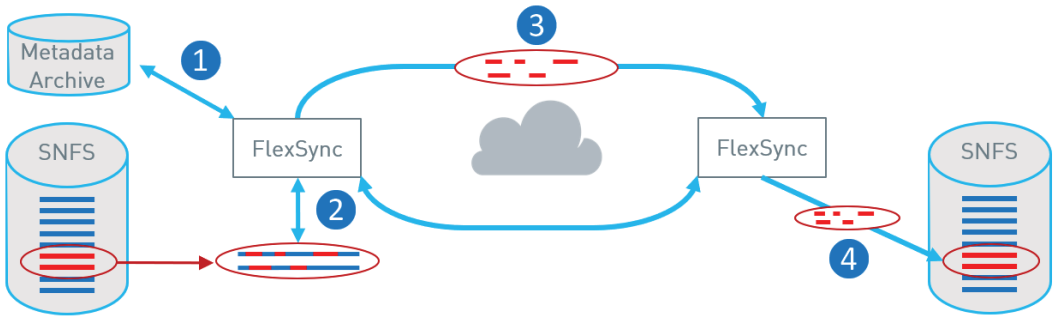
FlexSync may be used to replicate a single directory, an entire file system, or anything in between. Typical uses include replication of an entire file system for DR protection, and replication of one or more directories to multiple remote sites for content distribution.

Several features of FlexSync's design contribute to its performance and scalability. Most tools for replication rely on file system scanning to detect changes, which is inefficient, slow, and resource intensive. For very large file systems, scans may take days to finish, unacceptably hurting performance. FlexSync leverages StorNext's patented MD Archive to detect changes, using a simple database query instead of a resource-hogging walk of the file tree.

Once changed files are discovered by querying MD Archive, only the changed blocks of the changed files are sent between the source and destination, using a technique known as delta block compression – as shown in Figure 23. This minimizes the amount of network bandwidth required for synchronizing changes. Files are fully reconstituted on the destination system as changes arrive.

FlexSync takes advantage of multi-threading and multi-streaming when replicating data, even within a single task. In order to scale replication capacity, FlexSync data movers may be deployed on multiple nodes in the cluster.

Synchronization activity may be scheduled or manually initiated and is easily monitored with a dashboard GUI.



No.	Action
1	Query Metadata Archive for new or changed files on source
2	Compare block checksums, identify new or changed blocks
3	Send new and changed blocks only to target
4	Changes merged into target SNFS

Figure 23 – FlexSync Delta Block Compression

Beginning with StorNext v7.1, FlexSync may also be used to synchronize content residing in a StorNext file system with a bucket on S3-compatible object storage, for example in a public cloud. This opens exciting new capabilities and workflow possibilities for StorNext.

In the simplest use case an administrator now has the ability to quickly toss a copy of some specific content up to the cloud for safekeeping, and keep that copy up to date either manually or automatically. This is true even for unmanaged file systems that don't have the option to use FlexTier to store files in the cloud. Enabling version retention on the S3 repository provides simple cloud-based DR that protects files against innocent accidents and malicious actions like ransomware.

The real power of FlexSync's cloud capability is revealed when multiple sites are involved. Because both file data and file system metadata are synchronized to the S3 destination, data can be synchronized up to the cloud by one StorNext site, and synchronized down from the cloud to one or more additional StorNext sites. This enables use cases from simple publish-subscribe data distribution to "follow the sun" tag-team production to a global, shared, cloud-based content repository. Multiple options are provided for conflict resolution in the event conflicting writes are made to the same content, and versioning enables easy access to previous file versions.

IMPORT / EXPORT

Data, especially when it is fresh and new, doesn't often sit still. As the word implies, "workflows" are all about information / data / files moving through a process akin to a digital manufacturing line. Source content is acquired, various transformations and modifications to that content are made, and the result is a digital product such as the next superhero movie, a 3-D scan of a heart muscle, or a virtual tour of a planned building. Teams in multiple locations, the use of cloud-based resources, file migrations, 3rd-party data interchange needs and other factors may require that files move into and out of a storage system, not just move around within it.

StorNext addresses this reality with a variety of import/export capabilities. Of course, at any time files may be simply copied out of the file system to somewhere else. This works for small amounts of data and occasional use. StorNext's import/export capabilities shine when sizeable amounts of data must be moved on a regular basis.

Tape

Exporting files on tape is good for moving large amounts of data between instances of StorNext or between StorNext and other systems. This is useful when moving flies between far-flung sites, producing copies for archiving by third parties (business partners, regulatory agencies, cultural institutions, etc.), or delivering files to a customer while completely removing them from StorNext, as is often required with contract work.

For transferring files between StorNext systems, tape media formatted as ANTF is used. ANTF is StorNext's native tape format and is optimized for performance and efficiency. For transferring files to non-StorNext systems, or when the ultimate user of the media is unknown, media may be exported in the industry standard LTFS format. For either media format, manifest files are optionally created as part of the

export and may be used to accelerate later import. A separate report file may be created and used as a human-readable record of the export contents or as the basis for creating an import batch file.

The export process can export all files on selected source media, or selected files specified individually or via a batch file. The original data may be exported, or a copy. As with all StorNext functions, CLI commands are available in addition to the GUI, facilitating custom scripting and automation.

To provide security and protect data integrity, tape encryption and the use of WORM tape media is supported.

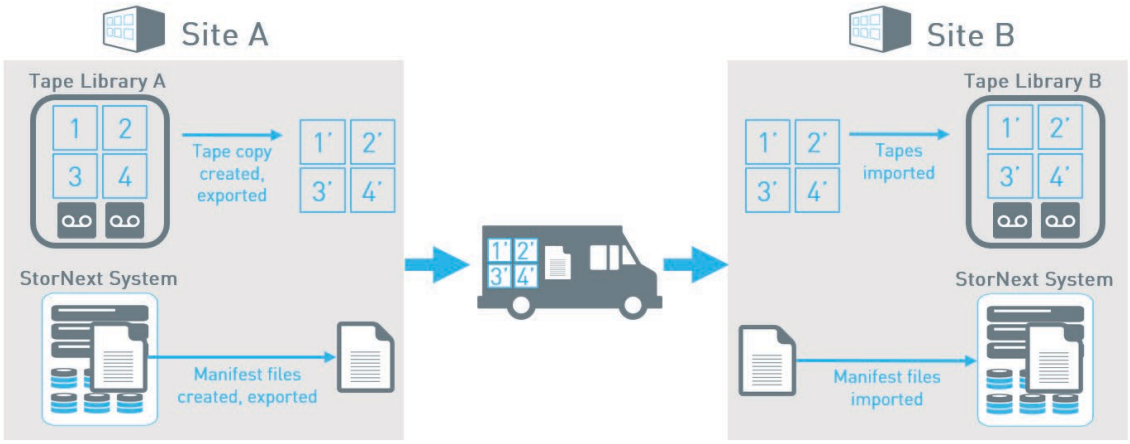


Figure 24 – Tape Copy Export / Import Flow

The import process enables files on ANTF or LTFS-formatted tapes to be quickly brought into another StorNext system. As with export, full media may be imported, or only specific files specified individually or in a batch file. Import actions use one of two modes—media import mode or file import mode.

With media import mode, tapes are permanently imported into StorNext, populating the database with knowledge of the files on the tapes. The destination directory is then populated with truncated files corresponding to the files on tape. File data blocks are retrieved from tape only when they are requested. This conserves space on the primary file system storage and enables import oversubscription, where more data may be imported than there is free space on primary storage.

File import copies files from the import media into the selected destination directory, then removes the media from the system. File import mode is useful when tapes must be returned to the originator after importing files.

Media Import	File Import
File inodes & metadata only created on import	File inodes & metadata created on import, and data blocks copied to SNFS disk
Files appear in file system as truncated	Files data is present on disk – not truncated
Data blocks copied on file access = retrieval delay	No retrieval delay, data blocks already on disk
Tapes must remain in library	Tapes may be removed from library

Table 9 - Media Import vs. File Import

Object Storage

As noted in the FlexTier section above, StorNext is fully capable of sending data to cloud and private object storage destinations. But what if you have a need to bring data from an object store into StorNext? There are many reasons to do so.

- Ingesting files shared by others in the cloud
- Capturing the results of data processing done in the cloud
- Permanently migrating files from an aging on-prem object store into StorNext
- Making content in an existing object store available through StorNext without migrating it
- Exfiltrating data from the cloud for better access and lower cost
- Etc.

Object import utilizes FlexTier's ability to connect to object storage systems, but to bring data in instead of sending it out. Because it uses FlexTier's infrastructure, any object storage destination supported by FlexTier is also supported by object import.

Like tape import, object import supports both file import and media import modes. For file imports, the contents of the source container will be copied into a destination directory on SNFS. There are options to specify how to deal with naming collisions. Once the import has completed, the source container may be disconnected from StorNext if desired.

For media imports, the source container is scanned and inode information is created in SNFS for each object found. Each object appears in the file system as a truncated file and is immediately accessible to StorNext clients. As with any truncated file, the data blocks are recalled on demand. Of course, the object storage container needs to remain connected for this to work.

Also, like tape import, the entire contents of an object storage container may be imported, or only specified objects. A list function generates a list of objects that may be modified and used to direct an import.

Objects are not files, so the method object import uses to map objects into the file system namespace requires some explanation. Object store containers are flat namespaces, while file systems are hierarchical structures of folders and subfolders. Objects may be stored in an object store with cryptic, one-dimensional names such as "`FIOVSNERH346HSA0`", or with names that look like (but are not) paths, such as "`/images/Alaska/bearsfishing.jpg`."

If the objects in a container to be imported have one-dimensional names, object import will create one file per object in a single flat destination directory, with each file name equal to the source object name. If the object names contain a defined delimiter such as forward slash ("`/`"), object import will create a folder structure that mirrors the path implied in the object name. For the example above, the import process would create a folder `/images` with a subfolder `/Alaska` containing a file named `bearsfishing.jpg`. Forward slash is the default delimiter, but any character may be specified.

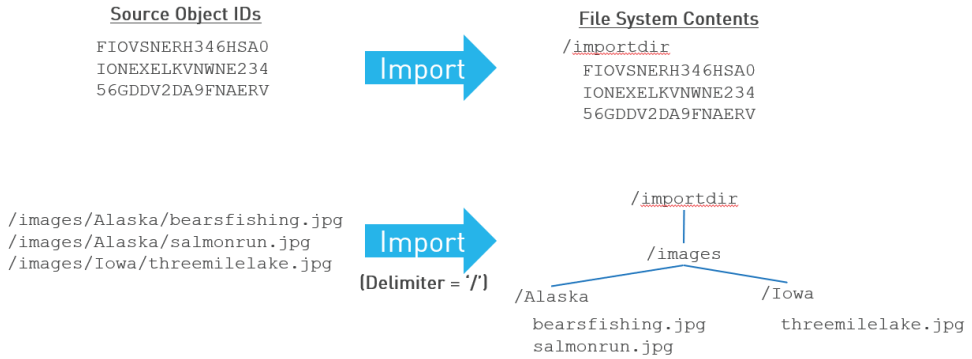


Figure 25 – Object Import Using a Delimiter Character

Two import filters are supported, time and prefix. The time filter enables restricting an import to only objects created after the specified time. A container can therefore be "watched," or periodically scanned for new content, which may then be imported. The prefix filter enables restricting an import to only objects with names beginning with a specified prefix. Using the example above, specifying a prefix of "/images/Alaska/" would import all objects with that prefix, including bearsfishing.jpg.

Archive Conversion

There are many reasons organizations contemplate changing their archive software. They may not be happy with some aspects of their current product or the way it is supported or priced. Or they may simply outgrow its capabilities and need something more capable and scalable. The biggest hurdle to migrating from one archive product to another is the process of getting data out of the old system and into the new, while maintaining access to the content. Like trying to change the tires on a moving car, it's difficult to do this non-disruptively. Transcribing hundreds of TB or PB of content can take months or years. During the conversion users must juggle two systems, and management gets to foot the bills for licensing and support of both old and new products.

Sometimes there is a better option. What if FlexTier could simply import the database of the existing archive product and "adopt" the media on which the content already resides? The conversion would be over very quickly, and no lengthy transcription would be necessary. This is precisely what the StorNext Archive Conversion service delivers. Legacy archives are moved behind StorNext's FlexTier feature, keeping the data accessible while integrating it with StorNext's high-performance storage architecture.

Because every customer has different needs and deployment details, archive conversion is offered as a custom professional services engagement. Even if a database import with media adoption isn't possible, the experienced Quantum professional services team can help customers navigate a more traditional migration of a legacy archive to sit behind StorNext in a way that minimizes headaches.

Management & Monitoring

Once they are initially configured, StorNext systems are relatively hands-off until workflow requirements change. As with any storage system however, administrators should keep an eye on capacity, performance, and the functioning of any data services that have been deployed. StorNext makes management and monitoring easy by providing a variety of tools, including a modern GUI, a traditional CLI, A web services API, and a cloud-based analytics system.

THE UNIFIED USER INTERFACE

Traditionally when deploying a file system such as StorNext that's independent of the underlying hardware, administrators must juggle multiple interfaces. At minimum there is one for each storage system (each disk array, for example), and one for the file system itself. This complicates management and impairs visibility. The Unified UI (UUI) was designed to solve this challenge. When StorNext is deployed with Quantum storage appliances (H-Series and F-Series, for example), both the storage devices and file system may be monitored and managed from the single UUI, including functions like storage expansion, NAS clustering, file system pools, replication, and more. If multiple StorNext systems are deployed for different teams or locations, they are all visible in the single UUI.

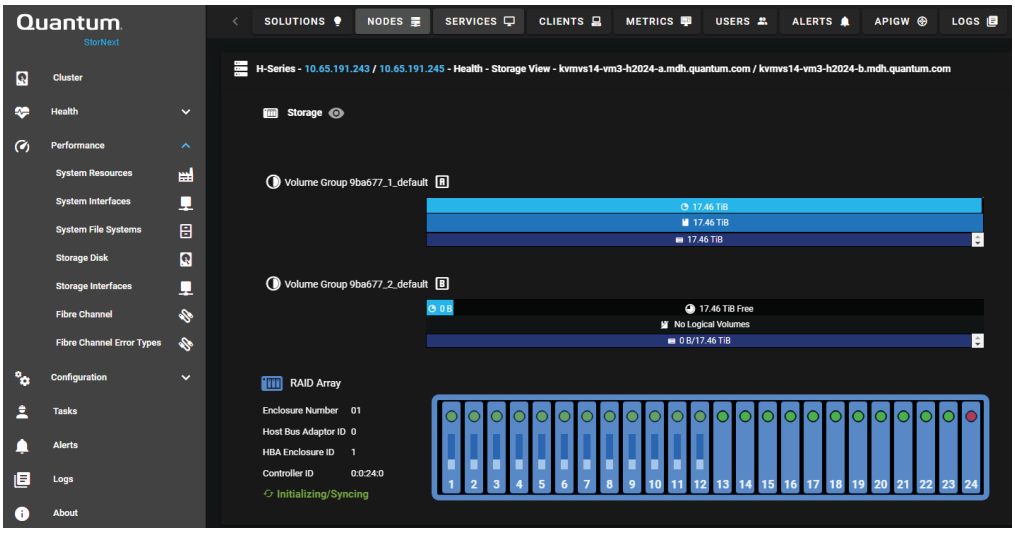


Figure 26 – StorNext Unified User Interface (UUI)

CLOUD-BASED ANALYTICS

As the sun sets on Los Angeles, it's rising on Mumbai. For an increasing number of organizations, "production" is a global affair, with teams on multiple continents contributing to the final product. The increased pace puts a strain on IT professionals, who must manage assets in multiple geographies and strive for 24x7 data availability. Quantum's Cloud-Based Analytics (CBA) software helps by offering centralized monitoring of Quantum hardware and software that's accessible from anywhere. Quantum products securely connect to the central CBA hub, uploading telemetry and log files. Customers can keep an eye on all their Quantum systems in one place, from file, block, object, and tape devices to video surveillance appliances. When support is needed, Quantum's service team uses the information collected by CBA to quickly provide solutions. CBA is also the foundational technology that enables Quantum's Managed Service and Quantum-as-a-Service offerings.

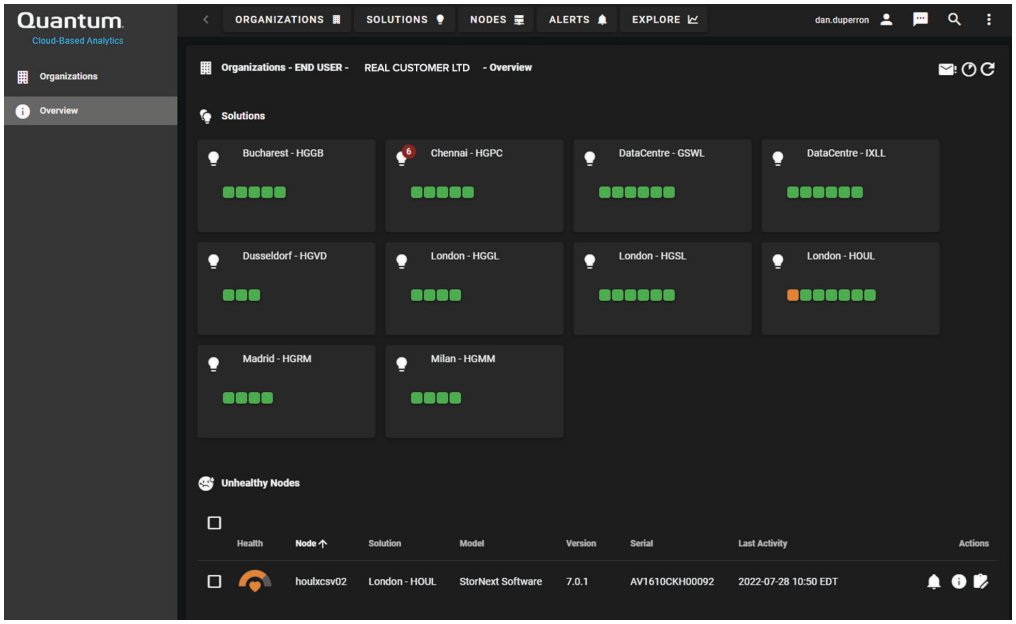


Figure 27 – Cloud-Based Analytics Organization View

COMMAND-LINE INTERFACE

Graphical interfaces are great, but there are situations where a good old command-line interface (CLI) is the most efficient way to get things done. CLIs can be particularly helpful when scripting common tasks or as "glue" for creating workflow automation systems. StorNext has a full featured CLI that supports every aspect of configuration and maintenance of the system. The CLI is fully documented in the StorNext Man Pages Reference Guide.

WEB SERVICES API

Every customer places unique demands on a storage system and integrates it into their operations in a unique way. StorNext functionality may be directed several ways. The GUI and CLI are good for admin-initiated ad-hoc tasks and configuration. Normal background processes use the built-in automation and scheduling capability. The CLI may be combined with custom scripting to automate routine tasks or sequences of events. For integrating with applications, a highly functional web services API is provided.

Many Quantum technology partners have integrated their applications with StorNext using web services. Via web services, an application can gain information about StorNext's internal state, such as whether a file has been stored to secondary storage and where, or even drive StorNext operations directly rather than using the default policy automation. A Media Asset Manager (MAM), for example, may direct StorNext to recall a set of files from the cloud at night when WAN usage is lower. The user simply interacts with their application (in this case the MAM), and the application controls any aspects of StorNext operations required.

Web Services API access is disabled by default. When enabled, it may be configured to use HTTP or HTTPS, and to require authentication (user ID and password) for requests, or not. Multiple web services users may be created, each with their own access rights. For the purposes of security, web services commands are divided into four categories. For each category, each web services user may have Read-Write access, Read-Only access, or no access (Disabled).

Security Category	Description
File Control	Used for all file-related web services calls
Destination Control	Used for all web services calls that deal with some form of media
System Control	Used for all system-related web services calls
Policy Control	Used for all policy-related web services calls

Table 10 - Web Services API Security Categories

The individual web services commands enable a great deal of status information to be collected and a lot of control exercised over the system. Functionally, the commands fall into several categories, summarized in Table 11.

Functional Category	Description
Archive	Return information about an archive, query an archive port, or change the state of an archive
Directory	Modify the class attributes of a directory or retrieve or recover files from media
Drive	Report or change the state of drive components and storage subsystems
File	Report on, retrieve, and store files to tiered storage
Job	Return information about jobs
Media	Manage media – copy, clean up, move, and report on
Object Storage	Report on Object Storage components
Policy	Manage and report on policies
Quota	Manage and report on quotas
Report	Return information about subsystem resource requests
Schedule	Manage and report on schedules
System	Get status of system and FlexTier components. Manage and report on backups.

Table 11 - Web Services API Command Functional Categories

A comprehensive guide to the StorNext web services API may be found here in the StorNext Documentation Center. It includes details on all available functions and options, tips for getting started, sample code in Java, Perl, and Python, and troubleshooting guidance.

SNMP & SMTP ALERTING

StorNext supports alerting via Simple Network Management Protocol (SNMP) and email (SMTP). SNMP configuration is a simple matter of enabling the feature and setting the community string and trap receiver. Email notifications are very configurable. Multiple email recipients may be defined, with custom settings for each based on the type of alerts desired (admin alerts, backup alerts, service tickets, alerts related to a specific policy class) and the severity (Low, Medium, High). Excessive alerting leads to “alert fatigue,” which can cause important notifications to be missed or ignored. The ability to customize and tune alerts enables recipients to see only notifications that are relevant to them, helping to ensure that alerts are taken seriously.

A Few Words About Security

In addition to the features already mentioned, (AD/LDAP integration, SEDs, ACLs, cross-platform permissions, checksums, encryption, WORM) there are other ways that Quantum ensures data security for customers running StorNext.

Beginning with StorNext 7, an additional tool named "snaccess" is available to provide administrators with positive and highly granular control over client access. Snaccess enables the definition of rules that determine which clients (by IP or subnet) may or may not access which file systems or paths, at which times of the day or week. These rules prevent unauthorized insiders from "share surfing" and gaining access to StorNext-hosted content, and the time of day restrictions are helpful for protecting content from changes during times when automated processing is occurring.

Upgrades for StorNext software and StorNext appliances have MD5 checksums calculated during the release process. The checksums are published and easily validated with publicly available tools to ensure the integrity of downloaded upgrades. For upgrades performed by Quantum personnel and service partners, validating the checksums is a standard part of any upgrade.

StorNext releases are tested with multiple commercial vulnerability scanners before being made generally available. Any issues found are investigated, and if necessary, remediated before code is released. Quantum will also work with customers who perform their own vulnerability scanning to help interpret and contextualize any 'hits.'

Quantum constantly monitors vulnerability databases published by CISA and other CSIRTs, evaluates newly identified security issues, and remediates as necessary. Customers and partners are notified proactively of critical issues and related fixes.

StorNext has been evaluated by KPMG and declared "TISAX ready", if deployed according to Quantum's TISAX recommendations. This means that StorNext may be used by organizations that must prove TISAX compliance.

Security is an important subject to Quantum. The StorNext roadmap contains additional security-related enhancements we are happy to discuss under NDA.

Conclusion

Whatever the need for unstructured file storage, chances are good that StorNext Scale-out File Storage can meet it. StorNext's unique combination of performance, scalability and flexibility makes it a natural choice for any organization struggling with an explosion of data that must be available, shared, protected, and stored cost-effectively.

Primary Author:

Dan Duperron

Technical Marketing Architect

Quantum Corporation

Quantum technology, software, and services provide the solutions that today's organizations need to make video and other unstructured data smarter – so their data works for them and not the other way around. With over 40 years of innovation, Quantum's end-to-end platform is uniquely equipped to orchestrate, protect, and enrich data across its lifecycle, providing enhanced intelligence and actionable insights. Leading organizations in cloud services, entertainment, government, research, education, transportation, and enterprise IT trust Quantum to bring their data to life, because data makes life better, safer, and smarter. Quantum is listed on Nasdaq (QMCO) and the Russell 2000® Index. For more information visit www.quantum.com.

www.quantum.com | 800-677-6268

WP00253A-v05 July 2023