FLASHARRAY™ FILE SERVICES

Native file systems extend FlashArray's applicability

FlashArray is well-established as the premier all-flash ***block*** storage system for enterprise data centers, presenting disk-like virtual *volumes* (LUNs) to host computers whose software file systems impose structure on stored data. Enthusiastic users have long wished that FlashArray performance, reliability, ease of use, and Evergreen™ immunity to product end of life were available for file data—home directories, project shares and so forth. To meet their needs, Pure Storage has developed *FlashArray File Services* (usually called simply *FA Files*), adding native file system capability to the FlashArray line. Architecturally, FA Files is a peer of the arrays' block volume services; both utilize the same underlying virtual and physical storage layers. This brief describes how FA Files extends the architecture that has made FlashArray the industry's benchmark for block storage to make the same qualities available for shared files.

FLASHARRAY FILE SERVICES

FA Files operates either alone or concurrently with FlashArray volume services. It supports the features that enterprises require from file servers:

- Support for up to 500 million files[1] in as many as 24 file systems
- Thin provisioning (occupies no space for "sparse" file byte ranges in which no client has ever written data)
- SMB (versions 1-3[2]) and concurrent NFS version 3 client access, including cross-protocol coordination of byte-range locks and client-side locking. The SMB implementation was developed entirely by Pure Storage engineers; it does not utilize any pre-existing or third-party technology
- Shares (exports) of different file system subtrees with different client access permissions
- Active Directory (AD) and Lightweight Directory Access Protocol (LDAP) authentication with Kerberos support for SMB clients, as well as NTLM for older Windows clients
- Ad hoc and automatic (policy-based) space-saving snapshots of data
- Quotas that restrict storage space consumption on a per-managed directory basis.[3]

[1] Smaller array models may have lower maximums.
With product management approval, larger arrays can support up to one billion files in selected applications.

[2] Use of SMB version 1 is discouraged due to security deficiencies that are remedied in later versions. Version 3 support is limited—transparent failover, multi-channel, and persistent handles are not supported.

[3] Managed directories are a unique FA Files feature. They are discussed starting on page 4.

When File Services is enabled on an array that also provides volume services, a portion of the array's primary controller resource is dedicated to it, but file and volume data are intermixed on flash. No per-file system provisioning or reservation of capacity is necessary, or indeed possible.

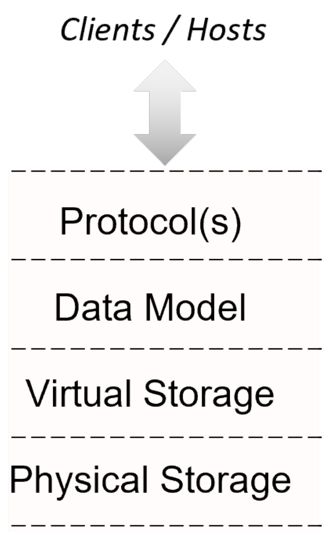

Figure 1: A Generic Model for Storage Systems

THE FA FILES ARCHITECTURE

The conceptual storage system software model in Figure 1 helps to understand how FA Files fits into FlashArray's Purity//FA software architecture. Whether a storage system presents blocks, files, or objects to clients, it typically contains the four layers shown in the figure:

Protocol(s)
Receives commands from and data written by client computers (usually called *hosts* in block storage contexts) and responds to reads with requested data

Data Model
Organizes stored data as files, volumes, objects, or other models. Implements data model semantics and manages the space presented by the virtual storage layer

Virtual Storage
Presents storage to the **Data Model,** typically virtualized to enhance I/O performance, data reliability, flexibility, or a combination of the three. Common techniques include caching, mirroring, striping, and RAID.

Physical Storage
Ultimately, systems store data in some form of persistent memory, the most common being flash, magnetic disk, and tape. This layer includes driver software that controls storage devices and transfers data to and from them.

MODELING FLASHARRAY

Figure 2 illustrates FlashArray's Purity//FA software architecture in the context of the model in Figure 1. In particular, the figure shows how Purity//FA provides coequal volume and file services:

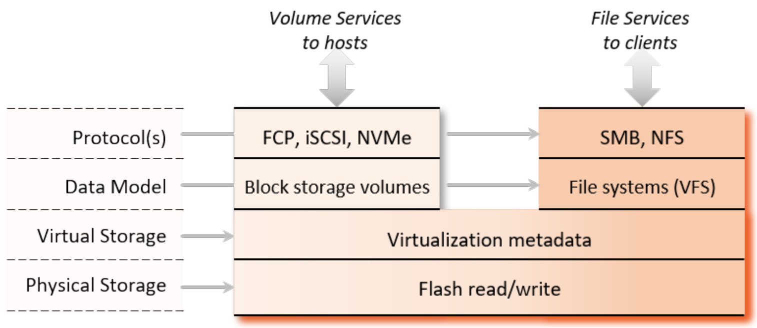

Figure 2: FlashArray's Purity//FA Architecture

Protocol(s)
At the protocol layer, Purity//FA Volume Services communicates with hosts using any of the FCP, iSCSI and NVMe protocols carried on Fibre Channel and Ethernet storage networks. File Services communicates with

clients using either the Server Message Block (SMB) or Network File System (NFS) protocol via TCP/IP connections on Ethernet networks.

Data Model

Volume Services presents disk-like virtual volumes to hosts, and controls host access, volume resizing, immutable atomic snapshots, and so forth.
File Services presents hierarchical file systems to clients via NFS and/or SMB. File Services controls client access, file system resizing, and immutable atomic snapshots.

Virtual Storage

Both Volume and File Services utilize the same virtual storage layer. Arrays present virtual storage in the form of metadata structures that they to locate data on physical storage. The structures allow identical content to be shared by volumes, files, and snapshots.

Physical Storage

Purity//FA allocates flash storage in *segments* distributed across multiple devices to optimize I/O performance and failure tolerance. To conserve physical storage space, the software *reduces* (globally deduplicates and compresses) incoming data before storing it. Multiple RAID checksums protect against data loss due to read failures.

INSIDE FA FILES

As Figure 2 illustrates, FA Files is a peer capability to FlashArray Volume Services. Both utilize the same virtual and physical storage substrate. Two key advantages of the common underpinnings are:

Identical Features for Volume and File Services

Volume and File Services organize data and store it persistently on flash in the same way, so the well-known features of FlashArray volumes—data reduction, automatic rebalancing, always-on encryption, recovery from read failures, eradication delays, and so forth—proven by nearly a decade of field experience in thousands of arrays, are also features of FA Files.

Simultaneous Volume and File Access

Most FlashArray models support volume and file services simultaneously. In many cases a FlashArray eliminates the need for separate file and block storage systems.

THE DATA MODEL LAYER

The FA Files **Data Model** implementation is a protocol-independent *Virtual File System* (**VFS**) that provides identical services to SMB and NFS clients. **VFS** achieves this through an intermediate **Data Store,** similar to a conventional object store, that communicates with the FlashArray Virtual Storage layer. The **Data Store** manages virtual storage for **VFS,** and provides the scalability and performance needed to support hundreds of millions of files in a system.

THE DATA STORE

In conventional file servers, file system software manages storage directly. The Purity//FA **Virtual** and **Physical Storage** layers are more versatile in that they reduce data, protect against read failures, and rebalance data automatically. The **VFS Data Store** isolates file system namespace management and semantics from direct interaction with storage.

The **Data Store** is a highly scalable key-value store that provides a single flat space of *items* (key-value pairs) with no awareness of file system hierarchies or semantics. **VFS** uses it to implement file system namespaces and to store both directory structures and file attributes and contents.

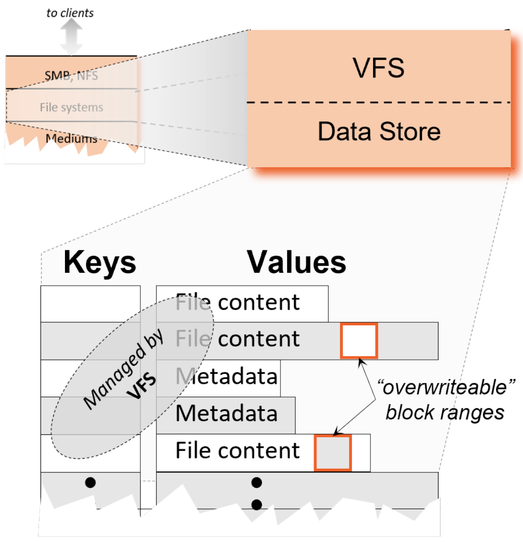

Figure 3: The FA Files Data Store

The **Data Store** resembles a conventional object store in that each key has a corresponding value that holds data and/or metadata. It differs, however, in that it supports overwriting ranges of bytes within a value, a property necessary to allow clients to overwrite byte ranges within a file. When a client writes data to a file, **VFS** overwrites only the blocks that include the range.[4] The storage layers append the overwritten blocks to the array's log and update metadata to reflect the data's new locations on flash.

[4] Purity//FA stores data in an append-only log that intermixes file and volume data, so "overwriting" is conceptual.

Managed Directories

VFS provides a protocol-neutral directory hierarchy **Data Model** for use by both SMB and NFS clients. Clients use POSIX or Windows APIs to manipulate directories in shares to which they have been granted access.

The client-visible file system roots created by array administrators are called *managed directories*. Administrators can also create up to three levels of subordinate managed directories under them. Managed directories differ from ordinary directories in that:

- Client-visible file system root and immediate descendant managed directories are the FA Files entities that administrators share with (export to) clients. The two lower levels of managed directories cannot be shared independently
- Only array administrators create and delete managed directories. They can only be deleted when they are empty and all shares have been removed or disabled
  As with conventional file systems, clients create and delete ordinary directories
- Administrators can assign quotas to managed directories to limit the amount of space they may consume. Space consumption is calculated on data as written by clients before reduction
- Administrators can attach *policies* to managed directories to control client access and specify snapshot schedules.
- Files and subtrees cannot be moved from one managed directory to another, nor can hardlinks cross managed directory boundaries[5]

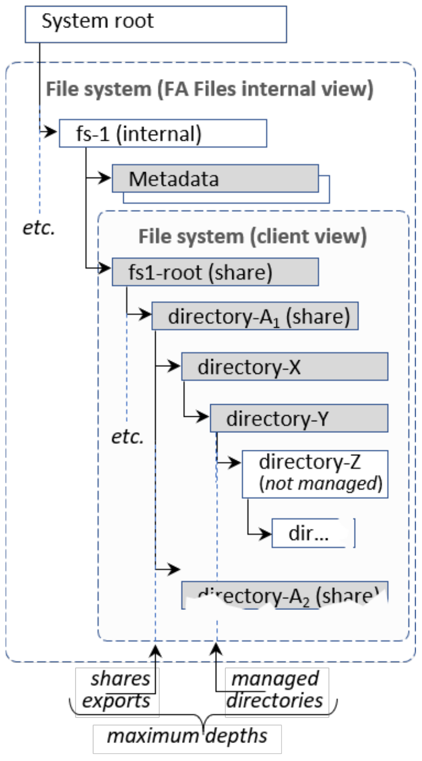

Figure 4: Managed Directories

A single file system can host thousands of managed directories. The top two levels of managed directories are the units in which data is shared with (exported to) clients. Sharing managed directories, each with its own client access and snapshot policy and quota, obviates the need for each application to have a separate file system in most cases. Managed directories are appropriate for data sets that are expected to remain on the same array throughout their lifetimes, whereas separate file systems may be preferable for unrelated data sets that may be treated differently in the future (e.g., migrated to different arrays). FA Files users may wish to consult their Pure Storage system engineers about data set organization.

[5] These restrictions may be relaxed at some future time.

The VFS Hierarchy

Internally, **VFS** uses the **Data Store** to implement a single system-wide hierarchy that is not exposed to clients. The hierarchy contains system-wide metadata, file systems, and per-file system metadata. Figure 5 illustrates its major elements:

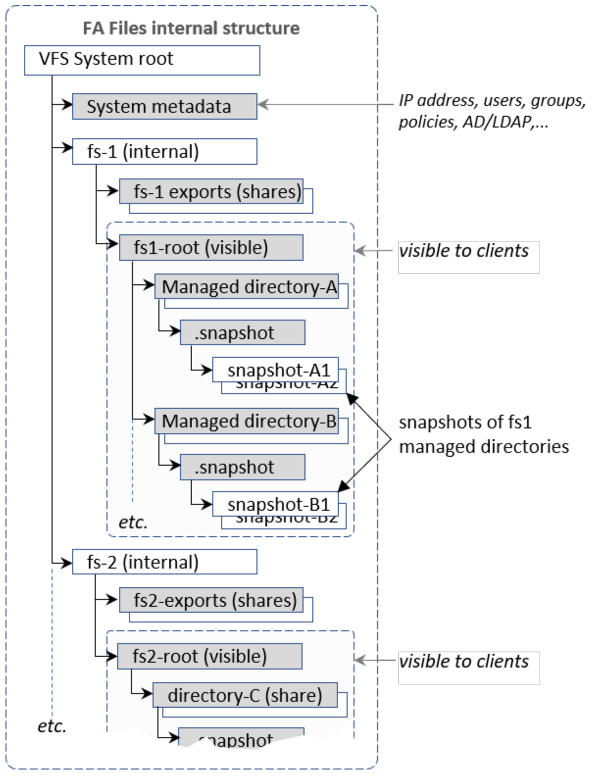

Figure 5: The VFS Internal Hierarchy

System root

Root of the **VFS** internal hierarchy. Not exposed to clients

Internal file system roots (fs1, fs2,...)

A **VFS**-internal root for each file system. Contains file system metadata (shares, snapshot policies, access permissions, etc.). Not exposed to clients

Client-visible roots (fs1-root, fs2-root,...)

Managed directories exposed to clients as file system roots. **VFS** creates these when array administrators create file systems. Administrators can create up to three levels of subordinate managed directories, the topmost of which can be exported

***.snapshot* directories**

Each share organizes its snapshots in a **.snapshot** directory. FA Files snapshots are immutable—while they exist, their contents cannot be altered. They are visible to clients as read-only descendants of the file system's **.snapshot** directory. Users can browse snapshots, read the files in them, and retrieve files for use by copying them to their original or other locations.

FILE SYSTEM STRUCTURAL INTEGRITY: ENVELOPES

All changes to the hierarchy shown in Figure 5, both internal to FA Files and visible to clients, are *transactional*. **VFS** guarantees atomicity for operations that affect file system structure, such as creating, appending, truncating, and removing files, and changing attributes, access permissions, and snapshot schedules. If an array fails while they are executing, they are reflected either in their entirety or not at all after restart.

Purity//FA makes structural changes transactional by writing sets of metadata objects called *envelopes* to the **Data Store**. Envelopes contain directories with lists of files, attributes, and other pertinent metadata as well as *journals*. Periodically, the software refreshes envelopes by applying accumulated journal entries to their contents. Using **Data Store** transactions to make journaling and envelope refresh atomic ensure the **VFS** hierarchy's structural integrity after recovery from array failures. After a restart, the hierarchy contains either pre- or post-update envelopes, but no partial updates that could result in structural inconsistencies.

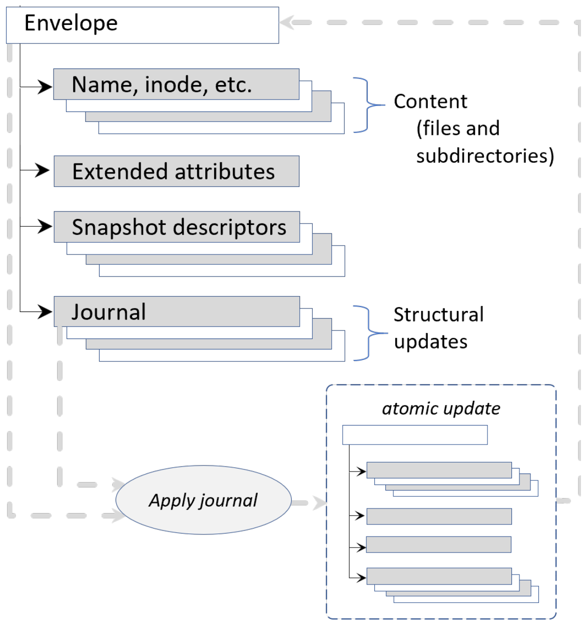

Figure 6: Envelopes

Envelopes protect the integrity of the hierarchy, but as with any network file system, client applications that use FA Files must provide for recovery from any failures of the server or client that occur while they have writes in progress.

STORING DATA AND METADATA

VFS uses **Data Store** keys to construct the file system hierarchy illustrated in Figure 5. Using the **Data Store** as an intermediary, it persists data and metadata on flash. Each **Data Store** item consists of a key and a value (data or metadata item) of arbitrary size.[6] The **Data Store** uses **Virtual Storage** layer services to write items on flash and retrieve them on request.

Each file managed by **VFS** is represented as a **Data Store** item. The **Data Store** manages files similarly to the way in which FlashArray Volume Services manages volumes, using common metadata structures to map virtual addresses to the flash location(s) of current data. With this

[6] FA Files supports files of up to 64 terabytes in size.

approach, Purity//FA provides the same performance, resiliency, and data reduction for both files and volumes.[7]

The **Physical Storage** layer allocates flash in which it stores volume and file data from a single free space pool. Like volumes, therefore, files are inherently *thin-provisioned*—aside from the metadata that describes them, they occupy physical space only to contain the data that clients write to them. The **Physical Storage** layer stores file and volume data written by clients and metadata in a single log in approximate order of arrival.

SECURING FLASHARRAY FILES

FA Files controls client and user access to data and protects data against both misappropriation and loss due to user and administrator errors.

CONTROLLING CLIENT AND USER ACCESS

FlashArray administrators limit access to FA Files services to specific subsets of an array's Ethernet ports with *virtual network interfaces* (**VIF**s) through which an array presents shares.

Arrays connect to any of LDAP (for NFS), NTLM (for older Windows servers), or AD (for NFS and SMB) servers to authenticate individual users before granting access to shares for which array administrators have authorized them. Alternatively, arrays can manage authentication locally.

PROTECTING FILE DATA AGAINST MISAPPROPRIATION

Purity//FA's **Physical Storage** layer applies AES-256 encryption to all data and metadata before writing it on flash. Encryption is "always-on"—it is not an option. Thus, even if flash devices were removed from an array and could somehow be read, data would not be exposed. The software manages data encryption keys internally; it never exposes them on any external interface.

For situations in which network security is questionable, FA Files supports SMB's *on-the-wire* data encryption. SMB clients encrypt data before sending it to the array; the FA Files **Protocol** layer decrypts it prior to processing, to allow deduplication and compression. The **Physical Storage** layer encrypts all file and volume data after reducing it but before staging it in NVRAM or writing it on flash. Thus, encrypting data on the path between SMB clients and FlashArrays secures it from point of origin to retrieval without sacrificing the benefit of data reduction.

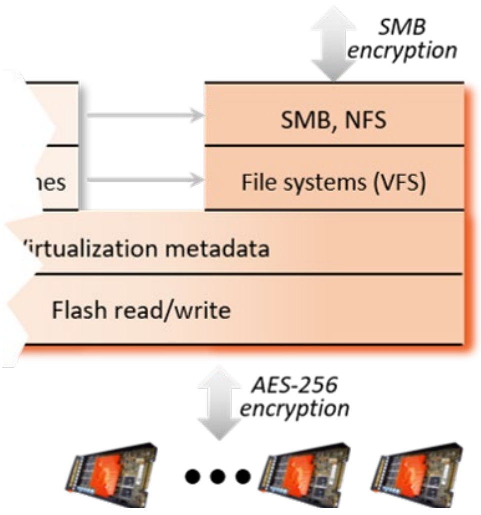

Figure 7: SMB "on the wire" Encryption

[7] That a software architecture created to support hundreds of volumes scales to hundreds of millions of files is a testament to its versatility.

PROTECTING DATA AGAINST USER AND ADMINISTRATOR ERRORS

FlashArrays provide two facilities that protect against file and volume data loss due to user, administrator, and application errors:

Snapshots

Purity//FA takes *point-in-time* (logically instantaneous) snapshots of managed directories, either automatically on administrator-specified schedules or on administrator command. Snapshots are effectively read-only versions of managed directory contents at specific points in time. They can be destroyed by an array administrator, but their contents cannot be altered. They are visible to clients as descendants of the `.snapshot` subdirectories of managed directories. To restore data items to a point in time prior to a data loss or corruption event, a user would delete the corrupt items from the live file system and replace them by copying content from the appropriate snapshot.

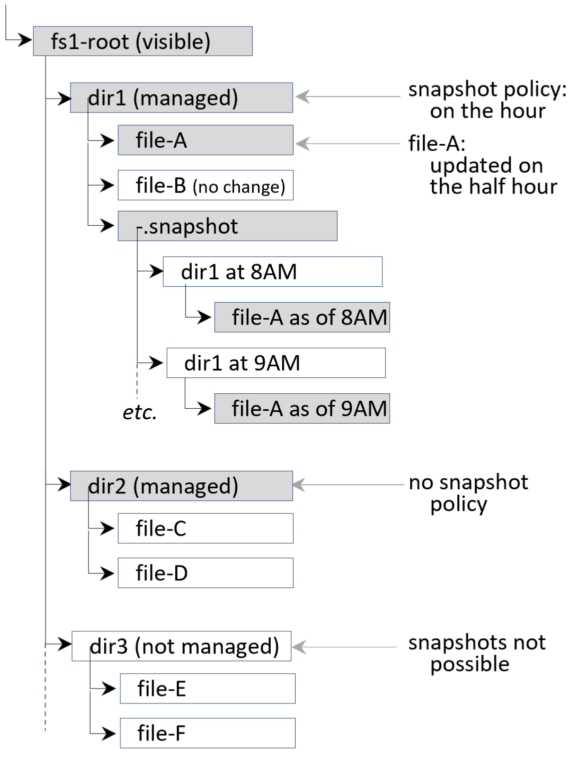

Figure 8: Snapshots of Managed Directories

Snapshots consume physical storage only when clients alter data in the managed directories on which they are based. Typically, administrators automate snapshot creation by attaching *policies* that specify frequency and retention to managed directories. Snapshots can be scheduled as often as every five minutes. Purity//FA automatically destroys them after the retention times specified in the policies that govern them.

Eradication delays

When an array administrator *destroys*[8] a snapshot, it becomes inaccessible to clients immediately, but it is recoverable for 24 hours before Purity//FA eradicates its contents. During that time, an administrator can restore it for client use. If physical space is urgently required, an administrator can *eradicate* destroyed snapshots, commencing background reclamation of the storage they occupy. Once a snapshot is eradicated, whether through lapse of the 24-hour grace period or by administrator command, it cannot be restored.

[8] Pure Storage CLI commands use the term **`destroy`** to direct removal of objects that contain data stored by users. For other objects, it uses the more conventional **`delete`**.

SafeMode

Snapshots can help protect against ransomware and other forms of malware by providing valid copies of data that attackers have encrypted or corrupted. To prevent attackers from destroying snapshots, users can enable *SafeMode* for enhanced protection. When SafeMode is enabled, live cooperation between Pure Storage Support engineers and designated trusted customer representatives is required to destroy snapshots or alter schedules.

USING FA FILES

All but the smallest current FlashArray models support FA Files.[9] Readers are advised to consult with Pure Storage representatives to identify array models to meet their file and volume storage and performance requirements.

Controlling Client Access to File Data

Client computers connect to FA Files shares via **VIF**s that include some or all of an array's Ethernet ports. Transparent failover and non-disruptive upgrade require that each VIF include one or more ports on each of an array's controllers.

Array administrators can create *access policies* that specify rules for client access and associate the policies with shares. Access policies are independent FA Files objects. Any policy can be associated with any share. Administrators manage access policies by adding and removing rules that specify client access permissions and by enabling and disabling entire policies. Modifications to a policy immediately affect all shares with which it is associated.

Accessing Files via SMB and NFS

FA Files is not a single-protocol implementation with add-on support for a second protocol. It uses **VFS** to read, write, and manage files whether clients access them via SMB or NFSv3. The **Protocol** layer adheres to SMB and NFSv3 locking rules and blocks attempts by clients using NFSv3 to access byte ranges locked by SMB. However, because NFSv3 is designed to be stateless, issues may arise when using NFSv3 to access data used by applications that rely on SMB state. Two significant examples are:

NFSv3 access to files that are 'open exclusive' by SMB
Because NFSv3 does not have an explicit file open operation, it does not respect SMB's open exclusive state.

NFSv3 client access to data with SMB oplocks
NFSv3 client accesses to data within the scope of an SMB client's oplock generate break notifications to the SMB client per the protocol specification. Clients (typically file system

[9] Some of the smaller models do not support FA Files and FlashArray Volume Services concurrently. Readers are advised to consult Pure Storage representatives for specific capabilities.

drivers) degrade or release oplocks as appropriate to the circumstances. This is usually transparent to applications.

For these and similar reasons, Pure Storage suggests disabling NFSv3 access to any shares used by applications that rely on SMB locks for correct operation.

QUOTAS

Array administrators can assign quotas to managed directories[10] to limit the amount of data that clients may store in them. Quota limits apply to data as written by clients, prior to reduction by arrays.

FA Files supports *nested* quotas. An administrator can assign a quota to a file system[11] and to managed directories within it. Administrators can overcommit a file system's overall quota using nested quotas, but the file system quota supersedes lower-level ones.

For example, a file system and each managed directory within it can all have 100GB quotas. In this scenario, the file system is limited to 100GB, which can be consumed by its managed directories in any way. A single directory could consume all 100GB; two directories could consume 70GB and 30GB respectively, and so forth, but the total consumed by all managed directories cannot exceed the file system's 100GB.

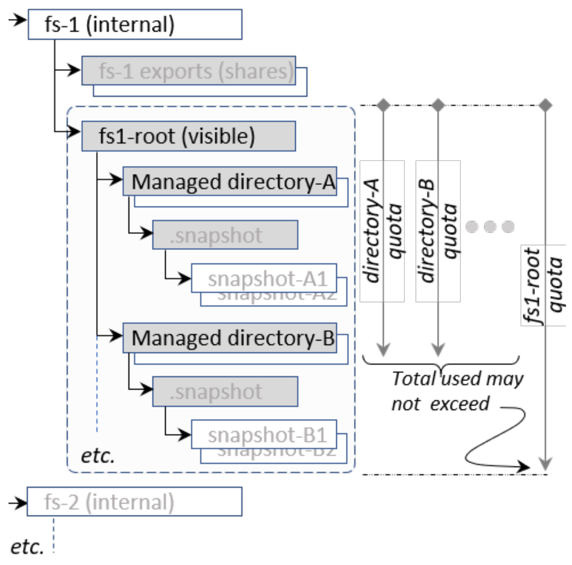

Figure 9: Nested Quotas

The CLI and GUI report available space for subordinate managed directories in the context of the overall file system's quota. In the above example, if each of two child directories contains 30GB, their available space is reported as 70GB, even though their assigned quotas are 100GB. As one of the child directories receives more data, the available space of the other decreases so that the file system's overall quota is honored.

Managed directory ownership is controlled by array administrators. FA Files generates an alert to a managed directory's owner when the directory's storage consumption reaches 80% of its quota, and a second alert when consumption reaches 90%. Alerts can be directed to users, groups, or both. Thus, for example, a managed directory that represents a project can have alerts sent to the project's IT manager or to all members of the project group. Managed subdirectories within it can be owned by individual users, and alerts related to the subdirectories directed only to them.

[10] Quotas cannot be assigned to regular (unmanaged) directories.

[11] From a **VFS** standpoint, a client-visible file system *is* a managed directory.

Storage Allocation and Array Capacity

FlashArray files and volumes are inherently thin-provisioned. Aside from a small amount of metadata that describes them, they consume physical storage only for data written by clients. Administrators and users do not reserve physical storage for specific volumes or files.

Purity//FA is designed to deliver full performance even when an array reports that its physical flash is 100% occupied. Administrators new to FlashArray may be surprised to discover arrays reporting that their storage is more than 100% occupied. Arrays achieve full performance at 100% occupancy by:

- Reserving a certain amount of physical capacity for the software's internal use. Arrays do not explicitly report reserved capacity
- *Throttling* (slowing down) writes when occupancy approaches 100% of reported capacity to allow time for software to free space occupied by obsolete data
- Generating alerts to array administrators and to Pure1 to provide "breathing room" for alleviating close-to-full situations by eradicating unneeded data or installing additional physical capacity.

When array occupancy is above 100% of reported capacity, client and host writes are "throttled" (delayed slightly) so they experience increased response times, but they do not fail abruptly (read performance is not materially affected). When the software frees enough space to reduce occupancy below 100% of reported capacity, it resumes full speed write execution.

High Availability and Failover

If a FlashArray primary controller fails, the array's secondary controller takes over its role. Each **VIF** used by FA Files must therefore include Ethernet ports on both controllers, so when failover occurs, the new primary controller responds to clients on the same virtual IP address. **VIF**s must also include ports on both array controllers for non-disruptive upgrade to be possible.

Monitoring File Services Performance

Administrators can monitor array performance history via the CLI and GUI. Performance history displays may include all access to an array or may be filtered to restrict displays to volume or file services only, to NFS or SMB access only, or to specific managed directories.

SUMMING UP FA FILES

FA Files expands FlashArray capabilities with native file support based on Purity//FA's **Virtual** and **Physical Storage** layers, proven by nearly a decade of service in an installed base of tens of thousands of arrays. **VFS** and the **Data Store** combine to provide robust, protocol-independent services to clients using either SMB or NFS to read and write file data. Both protocols have been developed entirely by Pure Storage engineers; they do not rely on previously existing packages such as Samba. Unlike file servers originally developed for one protocol and adapted to support a second, FA Files is strictly layered; both SMB and NFS protocols use the same **VFS** and **Virtual** and **Physical Storage** layers to a provide uniformly high quality of service to both client communities. For applications that require it, FA Files supports simultaneous client access via both protocols.

FA Files allows data centers to consolidate on premise storage for volume and file applications such as users' home directories and project directories in highly available, high-performing, cost-effective, space-efficient arrays that may contain up to 1.4 petabytes of physical flash.