

FLASHARRAY™ RESILIENCE

re-sil-i-ence

noun

1. *the ability of a substance or object to spring back into shape; elasticity.*

“nylon is excellent in wearability and resilience”

2. *the capacity to recover quickly from difficulties; toughness.*

“the often remarkable resilience of so many British institutions”

<https://www.google.com/search?q=resilience+definition>

The Pure Storage FlashArray architecture defines a family of enterprise-class storage arrays. A FlashArray consists of two controllers that store data persistently in flash-based *DirectFlash™ Modules (Solid State Devices)* in older and smaller arrays). Arrays present disk-like *volumes* to host computers over Fibre Channel, conventional Ethernet, or lossless RDMA Ethernet using the SCSI and NVMe protocols.

The FlashArray architecture is specifically designed for flash; it makes no provision for magnetic disk storage devices. Compared to disk-based arrays, FlashArrays offer significantly higher I/O performance, better power and space efficiency, administrative simplicity, and most importantly, *resilience* when things go wrong in an array or in its environment. This brief describes the role of FlashArrays in keeping data resilient when hardware and software failures, user errors, unauthorized access and theft, equipment and software upgrades, and data center disasters occur.

RESILIENCE IN CONTEXT

Today, “always on” 24x7 availability is an absolute requirement for data storage in the enterprise. Deployed FlashArrays deliver an average of 99.9999+% (“six nines”) uptime with no planned outages.¹ But availability isn’t just about keeping an array running—it’s about maintaining data integrity and sustained performance when failures occur, either in the array itself or in the environment in which it operates.

ARRAY RESILIENCE

FlashArray resilience starts with redundant hardware components and interconnects. Field-replaceable components, including entire controllers, can be replaced or upgraded without disrupting an array’s operation. Multiple ports in both of an array’s controllers make connection to two separate storage network fabrics possible for host-array path redundancy. Both controllers accept and respond to host commands on all ports, but one does all command execution. With this design, failover happens in seconds. Perhaps even more important, failure of a controller does not diminish an array’s I/O performance.

FlashArray’s Purity//FA software is layered on a Linux kernel that is streamlined to include only necessary functionality. It is designed to support continuous array operation, including *non-disruptive upgrades* (NDUs). New software versions can be installed and activated while an array is serving hosts.

Purity//FA cooperates with both array hardware and Pure1® to deliver best-in-class quality of service. Arrays report status and performance information to Pure1 every 30 seconds, and in addition, can be configured to send email and SNMP alerts for abnormal hardware conditions and for conditions such as low free space that may require administrative action. Pure1 archives the information it receives and analyzes it against a database of known potential issues. Where remedial or preventive action is indicated, Pure1 automatically create support cases and routes them directly to the company’s support organization. Most support issues are already being dealt with by the time an array owner becomes aware of them.

¹ Based on data continuously collected by the Pure1® cloud-based management platform.

DATA RESILIENCE

Purity//FA media organization and metadata structures are designed for robustness and data integrity. Arrays maintain all metadata and data in a combination of flash and NVRAM, including “in-flight” data that has not yet been stored on flash. Controllers are stateless, making failover rapid and enabling non-disruptive hardware replacement and software upgrades. Although it is not a common practice, an array’s entire complement of storage devices can be relocated to a different controller pair with no loss of configuration information or data.

FlashArrays implement RAID-HA technology, which supplements storage device read error detection and at a minimum, can recover data from any two concurrent media or device failures. Independent checksums on every segment of stored data localize the impact of recovering from media errors. Background tasks continuously read dormant data to detect and proactively correct latent media errors. When media rebuild is required, arrays distribute the load to avoid hot spots and minimize impact on host I/O.

“Always-on” software-based AES-256 encryption using specialized CPU instructions protects stored data against misappropriation of flash devices. Automatic transparent generation and management of encryption and device access keys makes decryption impossible without access to more than half an array’s devices. Arrays in physically insecure environments can be integrated with *Key Management Interoperability Protocol* (KMIP) servers or fitted with Rapid Data Locking (RDL) hardware with removable “smart cards,” either of which makes it impossible to decrypt data without access to them.

APPLICATION RESILIENCE

Purity//FA’s FlashRecover feature captures “point-in-time” snapshots of administrator-defined *protection groups* of volumes. Snapshots make it possible to “turn back the clock” of data to a known valid state (recovery point) when application errors or ransomware attacks occur. In addition, they are “frozen” images of data that can be used for backup, analytics, and testing and development.

FlashRecover snapshots share unmodified data with the volumes from which they are created, so initially they occupy almost no space. Over time, the space they consume is approximately equal to that occupied by overwritten data on the source volumes. Snapshot creation is fast—it is entirely feasible to capture an application’s data state as often as every few minutes.

FlashRecover snapshots are *immutable*. Once created, they do not change, and are visible only via administrative interfaces. They can, however, be “copied” to create *clones* (new volumes) for backup, analytics and testing, or to overwrite the contents of corrupted volumes for application recovery.

FlashArrays can play an active part in making IT operations resilient by periodically sending FlashRecover snapshots from a *source array* to one or more *target arrays* at remote locations. Remote replicas can make it possible to restart critical applications quickly with minimal data loss if a production data center becomes incapacitated. Target arrays at disaster recovery sites can replicate FlashRecover snapshots back to alternate volumes at source sites to speed return to normal operation after recovery from a primary data center outage.

FlashArray replication can also be used to distribute identical data from a central source array to as many as four target arrays. Conversely, many-to-one replication can be used to consolidate data from several arrays in a single central target.

Both local and replication FlashRecover snapshots are policy-driven—arrays take snapshots and replicate them at specified intervals and retain them for specified periods. Additionally, administrators can create on-demand snapshots and replicas at any time.

DATA CENTER RESILIENCE

FlashRecover periodic replication is suitable for most purposes. They may lag production data by a few minutes, but for most purposes, the lag is tolerable. Administrative simplicity and low overhead outweigh the need for up-to-the-second replication. For some applications, however, the value of update-by-update

replication of production data outweighs the cost of the IT resources it requires. Applications for which data loss would have a significant financial or operational impact need *synchronous* replication—every change to production data immediately reflected in an array at a safe distance from the production site. The primary purpose of such replication is disaster recovery—if a production data center is incapacitated, application instances at a recovery site can continue or resume service to clients quickly with no data loss.

The Purity//FA ActiveCluster™ feature maintains identical representations of administrator-defined *protection groups* of volumes on two interconnected FlashArrays. Unlike many conventional implementations, ActiveCluster is symmetric—there is no “source” or “master” array. With respect to replicated volumes, the arrays are peers. With ActiveCluster, *stretch clusters* of application servers in two locations separated by several kilometers can simultaneously process a data set that is physically represented at both locations.

DETAILS: ARRAY RESILIENCE

FlashArrays are based largely on common processor, DRAM, network, and flash hardware components. Historically, the price:performance ratios of these have improved as the technology within them has evolved, and so has that of the FlashArray family.

All FlashArray models are based on a common architecture; the differences lie in the type and number of components they contain. The smallest FlashArray//X10, the most powerful FlashArray//X90, and the largest capacity FlashArray//C60-1390 all consist of the same architectural building blocks:

Controllers (2 per array)

Interconnected servers that manage storage devices and interact with hosts over high-speed Fibre Channel or Ethernet (iSCSI or NVMe-oF) storage networks. FlashArray//X and FlashArray//C arrays house both controllers in a 3U-high chassis along with 10 or 20 storage devices and redundant power supplies and cooling fans.

Expansion storage shelves (0-2 per array—up to 4 SAS shelves)

Rack-mounted enclosures for storage devices used when more capacity is required than is available with the 20 devices that the controller chassis bay can accommodate. An array's first storage shelf connects to both controllers via redundant links as shown in Figure 1 on page 5).

Storage devices

Carriers containing DirectFlash Modules (DFMs—older and smaller arrays are equipped with SSDs) that mount in controller chassis and expansion storage shelves.

NVRAMs (2-4 per array)

Redundant staging areas accessible by both controllers used to stage data *in flight* (after a controller has acknowledged receiving it, but before it has been written persistently to flash). NVRAMs protect against data loss in the event of power loss or controller reset.

FlashArray is designed and built for lifetime continuous operation. All field replaceable units (FRUs), including controllers, and software versions can be replaced *non-disruptively* without interrupting host I/O. (Pure Storage Technical Support typically assists with complex replacements).

CONTROLLER HARDWARE REDUNDANCY

A FlashArray's two controllers are multi-core, multiprocessor computers whose PCIe buses are connected by a non-transparent bridge (NTB) used for heartbeating and message exchange. Both controllers use the NVMe protocol to communicate with DFMs and NVRAMs. Arrays equipped with SSDs use the SAS protocol to communicate with them.

Power supplies, cooling fans, storage network interface cards, interfaces to external storage shelves, and administrative network ports are all redundant. All field-replaceable units, including controllers, can be replaced while an array is operating.

CONTROLLER RESILIENCE

During normal operation, both FlashArray controllers receive and respond to host commands. The primary controller executes all commands, however. The secondary relays commands, responses, and data between hosts and the primary.

If an array's primary controller fails, or if an administrator switches controller roles, the secondary assumes the primary role. It takes control of flash devices and NVRAMs, completes any in-flight writes, and resumes service to serve hosts in a matter of seconds. Outstanding I/O commands that an array has not yet acknowledged at the time of a failover typically time out and are reissued by the host on paths to the new primary controller. Failovers automatically alert designated SNMP servers and email accounts, and contact Pure1, which automatically initiates any necessary remediation.

As long as hosts (a) support multipath I/O,² and (b) have connections to both controllers (both virtually always the case in well-managed data centers), a controller failure or removal from service leaves host access to data intact.

The primary-secondary controller architecture simplifies array design. Because controller failover is non-disruptive, highly reliable basic components such as CPUs, DRAMs, boot devices, and so forth, are not redundant. Arrays handle failures of these components as failure of the controller that contains them.

Another beneficial consequence of the primary-secondary design is that unlike “active-active” arrays in which multiple controllers process commands simultaneously, a FlashArray controller failure has little or no effect on array performance. The added processing power afforded by active-active designs is usually offset by the software complexity and increased latency that result from the need to coordinate access to storage. When a controller in an active-active array fails, overall performance decreases proportionally.

CONTROLLER “STATELESSNESS”

The primary-secondary array controller design works because FlashArray controllers are *stateless*. They store all configuration and operating state information persistently on flash devices accessible to both of them. Data stored on flash and staged in NVRAM includes metadata that makes it self-describing. When a failover occurs, the new primary controller recovers the array’s configuration (volumes, protection groups, hosts, host groups, connections, etc.) from flash, reads in-flight data from NVRAM, persists all write operations that have been acknowledged to hosts, and resumes execution of incoming host commands.

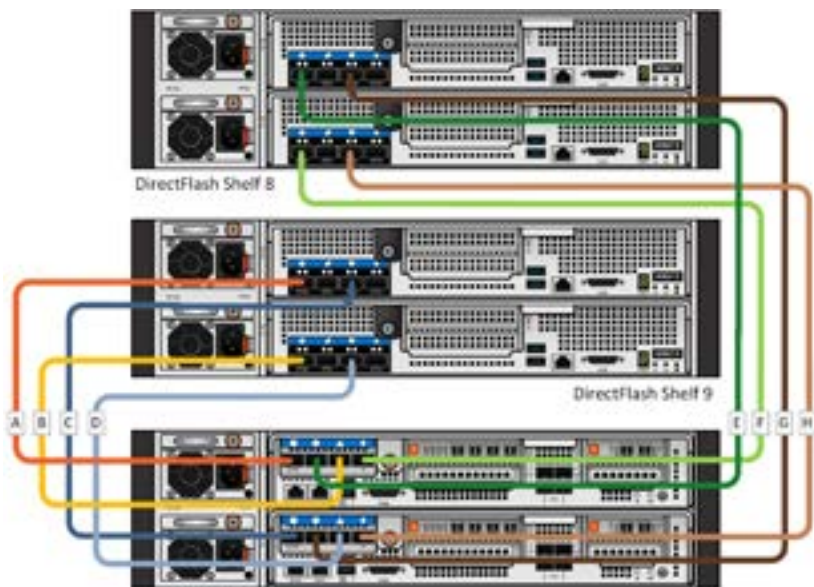


Figure 1: FlashArray//C Rear View Showing Shelf Connections

STORAGE SHELF RESILIENCE

Each FlashArray//X expansion storage shelf includes 28 device bays for mounting DFMs, redundant hot-swappable power and cooling modules and I/O interfaces. A passive mid-plane in the shelf chassis connects flash devices to both I/O modules. The two expansion shelves in a maximum FlashArray//C or FlashArray//X configuration connect directly to both controllers as shown in Figure 1, so the paths between flash devices in expansion shelves and array controllers are redundant.

Controllers use SCSI persistent group reservations to avoid the so-called “split brain” condition that can occur when a controller loses connection to its partner, and to establish quorum (the minimum number of flash devices required for an array to operate) when an array starts up.

² Host administrators should configure multipathing for *minimum queue length* scheduling to optimize I/O performance and preserve connectivity if there is a failure anywhere on a path between host and array.

HOST CONNECTION RESILIENCE

Each FlashArray//X controller has at least two Fibre Channel, 10GbE (iSCSI), or NVMe-oF storage network ports for connecting to hosts. Both controllers accept hosts' I/O requests, so a host can access any volume (LUN) or group of volumes via all array ports provided that:

- ▶ A storage network administrator has included the host and array ports in the same zone or subnet
- ▶ An array administrator has *connected* (permitted access between) the host and volume.

For completely redundant end-to-end connections, each controller and host should have storage network ports connected to two different storage networks.

NVRAMS AND RESILIENCE

FlashArray NVRAMs minimize write latency by *staging* (temporarily storing) incoming data until the primary controller reduces it and stores it persistently on flash. Arrays stage each incoming block in two NVRAMs for redundancy before they signal write completion to hosts. All further processing and storage occurs after the completion signal, so from a host perspective, write latency is consistently low.

Under normal circumstances, NVRAMs are write-only—primary controllers use DRAM buffers to reduce and write incoming data. Only when a controller restarts in the primary role after a failover or power outage does it read staged host writes from NVRAM,³ finish processing them, and write the data to flash. When a controller flushes a group of buffers (called a *segio*—see Figure 2 on page 8), each one to a separate device, the data in them becomes persistent and protected, so it frees the NVRAM staging space in the background.

If an NVRAM fails, Purity//FA generates an alert and immediately flushes all buffers that contain data staged in it. As long as two or more NVRAMs are functioning, the array continues to operate normally, including staging incoming data redundantly. If only one NVRAM is functioning, the software flushes all buffers that contain newly arriving data immediately after staging and acknowledging it. If *all* of an array's NVRAMs fail, the software stops accepting host writes but completes processing of already-staged ones. It continues to execute read commands.

FLASH DEVICE RESILIENCE

As flash memory technology evolves from MLC through TLC to QLC, Pure Storage continues to use consumer grade flash in its products to keep cost low, employing several hardware and software techniques to deliver reliable data integrity regardless of the type of flash employed.

The company improved flash device resilience significantly by developing its own DFMs to replace SSDs in its arrays. Not only can it choose the optimal vendor for raw flash, but with complete firmware control, it can optimize device behavior specifically for Purity//FA's needs, for example with array-wide wear leveling.

SSDs contain *overprovisioned* capacity that is invisible outside the device. The devices use it for “garbage collection,” to distribute write load, and as a reserve against blocks that become unreadable over time. DFMs expose all flash to controllers. Overprovisioning beyond quoted device capacity is sized to support a 5-year problem-free lifetime of continuous duty.⁴ DFMs minimize media wear by distributing internal writes across both exposed and overprovisioned flash. At the array level, Purity//FA distributes flash writes across all exposed capacity, including capacity reserved for array-wide garbage collection, further reducing flash wear.

Controllers query devices periodically to ascertain rate of overprovisioned storage consumption. When the available amount drops below a threshold, they generate alerts that suggest proactive device replacement.

³ FlashArray NVRAMs can retain staged data for at least three months without external power.

⁴ The company's experience with a growing installed base indicates that actual DFM (and older SSD) useful lifetimes are significantly longer.

IDENTIFICATION, AUTHENTICATION, AND QUORUM

Each flash device supplied by Pure Storage is initialized with a unique factory-written signature. Arrays will not recognize and use devices that do not have valid signatures. When a controller recognizes a device for the first time, it writes additional configuration information that identifies it as part of the array's *apartment* (complement of flash devices). At startup, Purity//FA confirms each device's signature and configuration. The software requires a *quorum* (minimum number of recognized devices) in order to operate.

The way Purity//FA uses an array's flash devices has two important consequences:

Capacity flexibility

FlashArrays allocate storage in fixed-size *allocation units* located on randomly-selected devices within a write group. Not all of an array's devices are required to have the same capacity. Array capacity can be increased by replacing devices one-by one with higher-capacity ones while it is operating.⁵

Location flexibility

Any initialized and configured device can be inserted in any bay in its array controller chassis or in an expansion shelf. There is no “wrong” location for a properly initialized and configured device—and no “right” one for a foreign device—an array will not accept a device that belongs to another array's apartment.

Location flexibility is particularly important in data centers with multiple arrays because it eliminates the possibility of data destruction when devices that contain data are erroneously connected to an array other than the one for which they were configured.

NON-DISRUPTIVE EVERYTHING

All field-replaceable FlashArray components are designed for live replacement, both in terms of operating continuity and in terms of sustained full performance. “Replacement” includes both repair and software and hardware upgrade. During an array lifetime of 10 or more years, all hardware components—NVRAMs, controllers, flash devices, and storage shelves—may be upgraded non-disruptively without service interruption or the need for data migration.

Purity//FA software upgrades are also non-disruptive. To upgrade an array's software, a technician halts the secondary controller, installs the new software on it, and reboots. When the secondary controller is up and running, the technician repeats the halt-and-install procedure on the primary controller. Halting the primary causes a failover to the (now upgraded) secondary. The upgraded controller reads NVRAM and completes any staged writes and metadata updates. Hosts configured with multipathing re-issue any unacknowledged commands on a path to the new primary controller.

⁵ Purity//FA does not utilize additional capacity until all devices in a write group have been replaced.

DETAILS: DATA RESILIENCE

Hardware's ability to survive component failures is only part of the resilience story. The ultimate objective of FlashArray is integrity of stored data—to return exactly what hosts have written when they retrieve it, regardless of what may have happened in the array or in its environment. Purity//FA goes to extraordinary lengths to ensure that the data hosts retrieve is exactly what was written.

PURITY//FA STORAGE ORGANIZATION AND DATA LAYOUT

Purity//FA organizes the flash in each of an array's devices in fixed-length blocks called *allocation units*. It allocates storage for writing and RAID-protecting data in *segments*—dynamically chosen groups of allocation units on separate devices as illustrated in Figure 2.

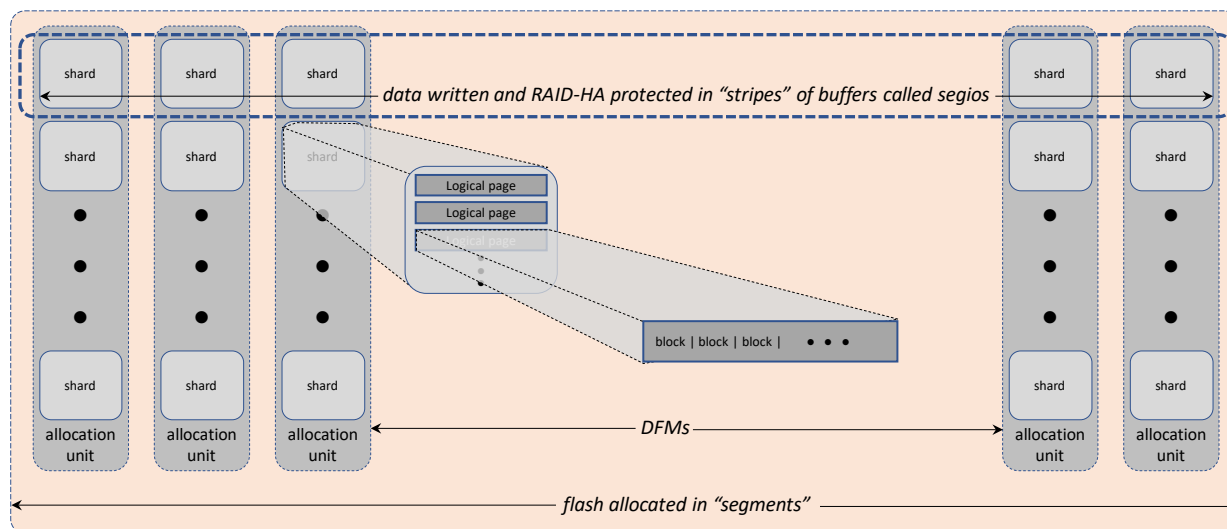


Figure 2: FlashArray Data Layout Showing RAID-HA Data Protection

The software chooses allocation units in a way that tends to balance I/O load (and therefore media wear) across an array's devices. Segments include space for both data and/or metadata and for RAID-HA checksums. Each allocation unit is organized as a set contiguous blocks called *shards* that correspond to the buffers from which the software writes data on flash. Stripes of shards are called *segios*.

Purity//FA *reduces* data as it is received from hosts by eliminating repeating byte patterns, deduplicating, and compressing it. Inline data reduction uses lightweight algorithms to minimize impact on host response. For *transient* data that is overwritten soon after it is created, inline reduction is the only reduction. Longer-lived data undergoes more thorough *deep reduction* in the background.

As the software reduces blocks of data, it packs them into stripes of shard buffers and calculates multiple RAID-HA checksums over them as they fill. With RAID-HA checksums, the software can reconstruct data when any two (and in some cases, even more) simultaneous read failures occur.

When a stripe of write buffers (Figure 2) is full, the software *flushes* (writes) them to the corresponding segio shards on flash. Segio writes are similar to conventional RAID “full-stripe” writes in that checksums are based entirely on buffer contents; no stored data is read for the calculations. Purity//FA splits each shard buffer into sections for writing so as to minimize any potential delays that lengthy writes might cause for unrelated reads, which are much faster.

Once all buffers of a segio have been written, an asynchronous software thread frees the NVRAM space occupied by the now-persisted data.

Purity//FA determines the number of shards in each segment it allocates based on the type of information it will contain and the number of devices in the selected write group of devices. The smallest possible

segment consists of an allocation unit for each RAID checksum and one for data. The largest consists of 28 allocation units on separate devices.

ENSURING THAT DATA IS RETRIEVED CORRECTLY FROM FLASH

In addition to RAID-HA, which verifies that flash pages are read correctly, Purity//FA calculates two other checksums to verify that data is read from flash correctly:

Logical page

As the software buffers reduced data and metadata, it calculates a checksum for each logical page. It uses page checksums to verify the contents of pages it reads and also that the intended pages were delivered.

Host block

The software calculates a checksum for every block of data written by a host prior to reduction and stores it with the block throughout its lifetime in the array. It uses block checksums as end-to-end verification for stored data.

If either checksum fails, the software uses conventional RAID rebuilding techniques to reconstruct the correct data and/or metadata.

ENSURING THAT DUPLICATES ARE REALLY DUPLICATES

FlashArray deduplication is alias-proof. When a sequence of incoming sector signatures matches those of an already-stored sequence, the software reads and compares the stored data to ensure that the two are actually identical before it processes the incoming as a duplicate.

RESERVED SPACE

Although page read errors are the most common flash failure mode, entire devices *can* fail, albeit infrequently. Purity//FA protects against device failure by reserving approximately 20% of an array's physical flash for internal use. Internal uses include garbage collection and rebuilding the contents of failed devices and unreadable pages. Each time the software allocates flash for writing data or metadata, it reserves the equivalent of two allocation units (see Figure 2 on page 8) in the write group to use in case unreadable contents must be rebuilt.

With distributed reserved space, the software can rebuild multiple units concurrently, thus minimizing the time to restore full protection after a device failure.

RESTORING PROTECTION AFTER FLASH DEVICE FAILURES

Failure of a FlashArray device reduces an array's storage capacity but does not degrade data protection. When a device fails, Purity//FA rebuilds the live data (only) in allocation units taken from reserved space, observing the RAID rule that no unit may be rebuilt on a device that already contributes a unit to its segment. To avoid bottlenecks, the software distributes units for rebuilding among the write group's other devices. It responds identically if another device fails while data from the first is being rebuilt.

Purity//FA begins rebuilding automatically upon detecting a device failure. No administrative action or prior configuration of spares is required. Rebuild time varies from a few tens of minutes to hours, depending on the amount of live data on the failed device and other load on the array. After rebuilding, data is fully-protected, but the array's physical storage capacity is less by that of the failed device.

When a failed device is replaced, its storage becomes eligible for allocation. Replacement devices may be larger than the ones they replace, but additional capacity only becomes available when an entire write group consists of the larger devices. Initially, new devices have a high probability of being selected during segment allocation due to their low occupancy and short in-service time.

DETAILS: MAKING APPLICATIONS RESILIENT

Prudent IT managers protect data against application bugs that corrupt it, as well as against operational errors such as unintended database table drops and file deletions. FlashRecover snapshots protect against both administrative and application errors. The FlashArray *eradication delay* feature provides additional protection against array administrator errors;

BACKGROUND: AGGREGATES OF VOLUMES

FlashArray volumes are virtual; the LBAs they present to hosts have no fixed relationship to the physical location or size of the data they contain. By default, arrays treat all volumes alike—there are no per-volume performance or availability properties. Because volumes share array resources equally, they can be configured entirely for application and administrative convenience.

FlashArray *host groups* and *protection groups* aggregate sets of related volumes to simplify administration of multi-volume and multi-host (e.g., cluster) applications:

Host group

A set of hosts (client computers identified by storage network addresses) connected to one or more volumes. Connecting a volume to a host group connects it to every host in the group. Adding a host to a host group connects it to all volumes connected to the group.

Protection group

A set of volumes specified either explicitly or by their connections to hosts or host groups. A single command (e.g., *snap*, *destroy*, *eradicate*) acts atomically on all volumes in a protection group. Volumes are added to or removed from protection groups as they are connected to or disconnected from hosts or host groups. Protection groups are the unit in which arrays replicate data to remote counterparts.

THE ERADICATION DELAY

Purity//FA provides a measure of protection against erroneous data destruction with a 24-hour *eradication delay* period that starts automatically when an administrator destroys a volume, protection group, or snapshot. Destroying makes data unusable immediately, but the array retains the destroyed object for 24 hours during which an array administrator can:

- ▶ Recover the object to its state at the time it was destroyed
- ▶ Explicitly *eradicate* the object immediately to start releasing the space it occupies for reallocation.

Once an object has been eradicated, either by command or by lapse of the eradication delay, its contents can no longer be recovered. The space it occupies is reclaimed by background tasks, so it becomes available for reallocation gradually, depending on I/O load and free space in the array.

SNAPSHOTS

FlashArray administrators can use snapshots to recover data from application and operational errors, effectively “turning back data’s clock” to a time that precedes the error. A snapshot captures an image of all volumes in a protection group at a single instant. Snapshots are immutable—they can be destroyed, but as long as they exist, their content does not change. To make their contents visible to hosts, an administrator *copies* them, either creating new volumes, or overwriting the contents of existing ones.

Initially a FlashArray snapshot shares all data with the volume from which it was created. As hosts overwrite data in the source volume, the array retains the overwritten data and associates it with the snapshot. Thus, the space occupied by a snapshot at any point in time is approximately that occupied by source volume data blocks that have been overwritten since snapshot creation. Because snapshots typically occupy little space, administrators often take them frequently—as often as every few minutes. An array can support thousands of active volume snapshots.

SNAPSHOT AUTOMATION WITH PROTECTION GROUPS

Protection groups help to automate regular creation and retention of snapshots, especially for applications that use multiple storage volumes. Administrators can specify separate schedule and retention policies for each protection group. The schedule specifies snapshot frequency; the retention policy specifies how long snapshots are kept. Retention policies include a duration for which all snapshots are kept and optionally, a subset of snapshots to be kept for an additional number of days. Figure 3 illustrates this with a simple example.

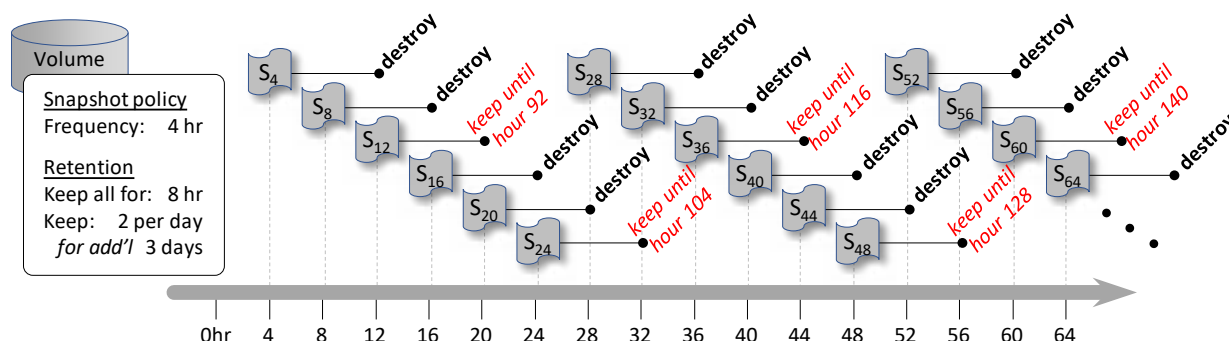


Figure 3: Sample Snapshot Schedule with Retention Policy

The policy illustrated in Figure 3 specifies a snapshot taken every 4 hours and retained for 8 hours. Thus, S_4 is taken at hour 4 and retained until hour 12, and so forth. Two of the six snapshots taken each day are retained for an additional 3 days (72 hours). Thus, rather than being destroyed at hour 20, S_{12} is retained until hour 92, and so forth.

With this policy, an application's data could be restored to its state as of:

- ▶ Any 4-hour recovery point within the most recent 8 hours
- ▶ Any 12-hour recovery point within any of the previous 3 days.

Administrators can also create *manual* snapshots, for example, to preserve data prior to one-time events such as application updates and anticipated host activity. Purity//FA retains manual snapshots until explicitly destroyed unless the administrator applies the protection group's retention policy when creating them.

As with volumes, destroying a snapshot starts a 24-hour eradication delay. At any time during the 24 hours, an administrator can recover a destroyed snapshot. After 24 hours, or on administrator command, snapshots are eradicated, and the space they occupy becomes eligible for background garbage collection. Once eradication begins, snapshots cannot be recovered.

Snapshots are not visible to hosts; to use a snapshot, an administrator copies it to a new or existing volume. Copying is virtual—Purity//FA creates or adjusts its metadata—but the new or overwritten volume shares data images with the snapshot until hosts write to it. To restore an application's data to a given recovery point, an array administrator copies the recovery point snapshot to the application's volumes. Copying automatically adjusts target volumes' presented sizes as necessary.

Any number of volumes can be created from a snapshot. Besides “turning back the clock” for live data, volumes created from snapshots can be used for point-in-time consistent backups, for application testing, or for data analysis. Volumes created from snapshots occupy only the space required to represent data that hosts write to them. They share unmodified data with the snapshots from which they are created (and with those snapshots' source volumes as well).

DETAILS: MAKING THE DATA CENTER RESILIENT

Enterprises that rely on their digital data to operate must anticipate disasters that incapacitate entire data centers and provide means to recover from them. They typically maintain physical or virtual IT recovery sites provisioned with sufficient IT equipment to run critical applications for some period. In addition to facilities and equipment, they must provide for:

- ▶ Up-to-date data at or accessible by the recovery site
- ▶ Resuming operation at the main data center once it is restored, again, with up-to-date data.

FlashArray asynchronous replication helps solve these problems. Replication reproduces protection group snapshots taken on a *source array* on one or more *target arrays*. When replication starts, the source array transmits the entire contents of the protection group's volumes to targets to establish a baseline. Thereafter, the source array transmits only the updates required to represent subsequent snapshots (sometimes referred to as *deltas*). Target arrays combine deltas with previous replicas to construct replicas of each snapshot received.

As with local snapshots, arrays take replication snapshots automatically on an administrator-specified schedule similar to the snapshot schedule example illustrated in Figure 3 on page 11. Snapshots taken as part of a replication schedule are not subject to local snapshot frequency and retention policies.

Source array administrators can explicitly enable and disable replication, which otherwise occurs automatically according to schedule. A policy may include a replication suspension interval, for example, to minimize network congestion during busy periods.

An array may be a replication target for more than one protection group. Data common to multiple groups may be shared, but each group is replicated on its own schedule, independent of others. On target arrays, replicas are identical to local snapshots. They can be copied to volumes for:

- ▶ Restarting applications at the target site after a source site disaster
- ▶ Use as sources for point-in-time backups using target array data images
- ▶ Use in data analysis performed on target rather than source array data.

As with local snapshots, creating volumes by copying snapshot replicas is nearly instantaneous, because new volumes share data with the replicas from which they are created until applications overwrite data.

FlashArray replication is secure. Source and target arrays must be explicitly connected by administrative action before replication can occur. A source array administrator obtains a connection key from each target administrator, and specifies the keys in commands that establish replication connections.

Finally, a target array administrator can control the space occupied by replicated data by allowing or disallowing replication for a protection group at any time. For example, an administrator can disallow replication if free space becomes dangerously low, and allow again it when the retention policy or other administrative action has freed sufficient space.

The volume snapshots that comprise a protection group replica can be copied to volumes on a target array and replicated back to the source array, where they appear as snapshots unrelated to the original source volumes. Reverse replication can expedite moving IT operations back to a primary data center that has been restored to a serviceable state.

24x7 APPLICATION CONTINUITY

There is a tension inherent in data replication. Replication imposes processing, network, and I/O overhead on IT production. Any replication strategy, application server or storage system-based, must balance the importance of up-to-the-minute replicas against its impact on IT production.

For most purposes, the key replication requirement is *coherence*—a replica that captures the state of an application’s data at a single instant. From coherent replicas, file systems and database managers can restore data to application consistency for resumed production, or for backup, reporting, and so forth.

Purity//FA snapshot replication overhead is low, and replicas are coherent. The state of replicated data may lag that of production data by a few minutes, but for most purposes, administrative simplicity and low overhead outweigh the need for update-by-update replication.

For some applications, however, the value of real-time replication outweighs the cost of the IT resources it requires. Where data loss would have a significant financial or operational impact, *synchronous* replication is a necessity—every change to production data must be immediately replicated at a safe distance from the production site. The primary purpose of synchronous replication is disaster recovery—if a data center is incapacitated, application instances at a recovery site can continue or resume service with no data loss.

SYNCHRONOUS REPLICATION BETWEEN FLASHARRAYS

Purity//FA *ActiveCluster* maintains identical representations of pods of volumes on two interconnected FlashArrays. Unlike many conventional implementations, by default *ActiveCluster* is symmetric—there is no “source” or “master” array; the two are peers with respect to replicated volumes.

Figure 4 illustrates the *ActiveCluster* concept. Peer arrays present *stretched* pods of volumes identically to hosts connected to both arrays. An array can host multiple stretched pods, each one replicated to a different peer. A mediator resolves any “split brain” situations in which peers cannot communicate with each other.

Both peer arrays execute commands addressed to volumes in a stretched pod, including SCSI extended copy and persistent group reservations.

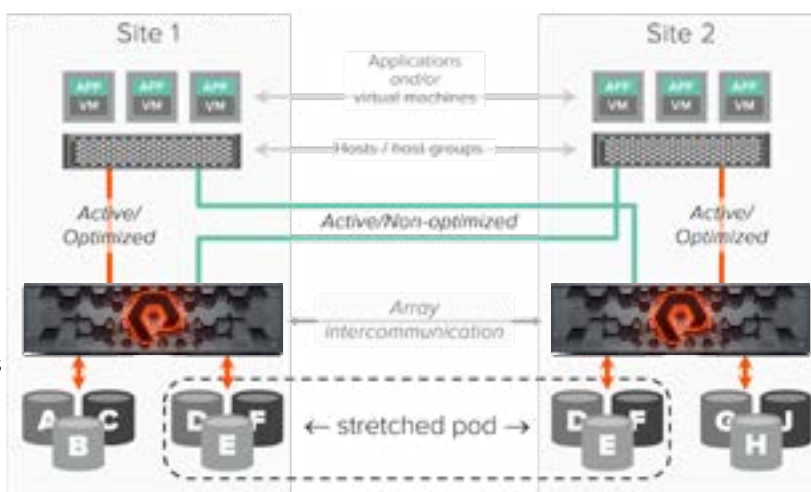


Figure 4: ActiveCluster Dual-Site Configuration

Both peer arrays’ administrators can manage the pod, volumes, protection groups (pgroups), and snapshots it contains. Either administrator can add, remove, and resize volumes, manage host-volume connections and snapshot schedules, and create clone volumes. All operations take effect immediately; both peer arrays represent pod volumes identically to all connected hosts, including contents, volume serial numbers, command ordering, and snapshots.

OPTIMIZED PATHS

Where all paths to pod volumes provide equal performance and reliability, the peer arrays are effectively additional host-volume I/O paths. For disaster protection, however, peer arrays are usually located at different sites, with local host-to-local array links and long-distance links to remote arrays. FlashArray administrators can configure host connections to stretched pod volumes to use *optimized* paths to the local array unless none are available.

ActiveCluster uses dedicated TCP connections to synchronize writes and configuration changes to stretched pods. If an element of the configuration fails:

Host failure

Host clustering facilities typically activate applications on backup hosts to take over processing load. Both arrays continue to operate, backup hosts route I/O commands to their optimized paths.

Inter-array communication failure

When peer arrays cannot communicate with each other, only one of them continues to execute stretched pod commands. The other immediately ceases to honor commands. A *mediator* provided transparently by Pure Storage in most instances determines which array remains active.

Array failure

If the peer of a failed array can communicate with the mediator, it remains active. Hosts route all pod I/O to the surviving array. During the outage, the active array executes commands from both local and remote hosts. When a failed peer array restarts, the peers resynchronize pod contents and resume synchronous replication. Hosts can revert to optimized I/O paths.⁶

Local storage network failure

For hosts with more than one optimized path to an array, multipathing software circumvents individual path failures. Failure of all optimized paths is equivalent to array failure. Hosts access pod volumes through non-optimized paths until optimized ones return to service. As long as inter-array communication is not interrupted, replication continues, so the arrays do not resynchronize.

While the primary application of ActiveCluster is protection against data center disasters, the technology can also be used to migrate live data between arrays while it is in use. Data may be migrated between arrays prior to planned data center maintenance, to replace an array with a more powerful one, as part of a data center relocation and so forth. In all cases, migration is non-disruptive to applications using the data.

⁶ Some host operating systems may require administrative action to reactivate optimized paths.

THE ULTIMATE IN RESILIENCE: PROACTIVE SERVICE

FlashArrays support troubleshooting mechanisms that accommodate a range of data center operating procedures. Troubleshooting is:

Automated

Arrays detect and report events and potential issues without human intervention.

Proactive

Most FlashArrays “phone home” performance and status information to Pure1 every 30 seconds. For potentially problematic events, Pure1 automatically notifies the Pure Storage Support organization, which begins remediation before issues become critical.

In most cases, troubleshooting starts with an array’s alert message to Pure1, and continues under the supervision of a designated technical support engineer. For data centers whose policies prohibit outside connections, arrays can direct event reports to internal email addresses, SNMP servers, and display them on an array console. In a few cases, onsite visits by a Technical Support engineer may be required.

SOURCES OF SERVICE INFORMATION

FlashArray automated troubleshooting uses three sources of information:

Array logs

FlashArrays log all potentially significant array events and activities such as administrative actions.

Frequent diagnostics

Every 30 seconds, arrays record performance and free space data, as well as hardware and operating status, and transmit the information to Pure1.

Alerts

Events that might affect array operation generate alerts and transmit them to one or more administrator-specified destinations for remedial action.

None of these include user data or information that would make user data identifiable.

ALERTS

Arrays generate and log alerts for events that could affect operation adversely (generally hardware or communication failures), or that administrators should heed (e.g., controller failover, low free space, etc.). They deliver alerts to:

Pure1

When phonehome is enabled, an array delivers logs, frequent diagnostics, and alerts to Pure1. When phonehome is disabled, arrays store logs for up to 10 days, and deliver them when it is re-enabled. Alerts not sent to Pure1 are delivered via email and/or SNMP.

Electronic mail

Arrays send actionable alert messages to an administrator-specified list of email addresses. Individual email addresses on the list can be temporarily disabled.

SNMP traps

Arrays send traps corresponding to alert messages to specified SNMP servers, with communication secured by administrator-supplied authentication parameters.

PROACTIVE REMEDIATION

Pure1 servers analyze the information received from arrays against a *fingerprint* database of known issues, and alert the Support organization about potentially problematic conditions specific to individual arrays for remediation before issues occur.

PURE1 REPORTING AND ACTION

Pure1 archives all information received from arrays and analyzes it for possible action. Logs, frequent diagnostics, and alerts provide historical background that makes it possible for Technical Support Engineers (TSEs) to detect array performance, hardware state, available space, and error rate trends.

Alerts are reported immediately so that timely action can be taken if necessary. The Pure1 alert router analyzes alerts and where warranted, generates notifications to:

Pure Storage Support

For alerts that indicate actual or potential issues that may require “hands on” attention. Where alerts indicate hardware failures, TSEs verify failure and initiate return material authorizations (RMAs).

The Pure Storage Software Reporting System

Purity//FA engineering analyzes software-related alerts for possible remedial action.

Pure Storage Sales and Application Engineers

For alerts such as “array close to maximum capacity,” that may require the attention of a Pure Storage representative.

An administrator can enable and disable phonehome at any time. All array communication with Pure1 is secured by TLS mutual authentication.

REMOTEASSIST

In some instances, the most efficient way to diagnose and service an array may be direct intervention by a Pure Storage TSE. The *Remoteassist* facility enables a remote Pure Storage technician to communicate securely with an array, establishing an administrative session for diagnosis and service.

Array administrators enable and control Remoteassist sessions. An administrator opens a secure channel between an array and Pure Storage Support, making it possible for a TSE to log in and administer the array. The array administrator can check session status and close the channel at any time. Arrays and Pure1 both log all commands issued during Remoteassist sessions, so a complete audit log of local and remote administrative and service activity is preserved.

FLASHARRAY RESILIENCE SUMMARIZED

For Pure Storage, resilience is about maintaining data integrity and delivering sustained performance when failures occur in the array itself, in applications that use it, and in the data center environment.

FlashArray resilience starts with fully redundant hardware that keeps an array running when components, up to and including an entire controller, fail. Controllers are “stateless,” so online replacement and software upgrade are possible. NVRAMs hold in-flight data to ensure that acknowledged writes complete after recovery from a controller failure.

All controller interconnects and controller-expansion storage shelf connections are redundant. Controllers present all volumes on all ports, so connecting controllers to two separate storage network fabrics makes end-to-end access to array data completely redundant.

FlashArrays protect against data loss due to device read failures at the individual storage segment level, optimized for the most likely flash failure modes. RAID-HA enables full data recovery from multiple simultaneous read failures on different devices. Arrays only rebuild live data affected by device failures. They rebuild affected segments automatically, distributing rebuilt allocation units among a write group’s devices for speed. When rebuilding finishes, data is fully protected, but the array’s capacity is reduced by the capacity of the failed device(s) until failed devices are replaced.

Logical page checksums detect read errors not reported by flash devices. Additional block checksums verify end-to-end correctness before arrays deliver data to satisfy host read commands. Deduplication is alias-proof—when Purity//FA identifies potential duplicates, it reads and compares the stored data to verify that it is in fact identical.

Arrays protect against administrator error with an overridable 24-hour eradication delays that begin when an administrator destroys a volume or snapshot. During the period, the administrator can recover a destroyed object or terminate the delay, for example if the space it occupies is urgently required.

Automated snapshots protect against application or administrative errors. Snapshots are space-saving—only changes to source volumes consume storage space—and immutable—administrators copy them to new or existing volumes for recovery, backup, data analysis, or other uses.

Replication of protection group snapshots to one or more remote target arrays makes data recoverable from disasters that incapacitate entire data centers. As with local snapshots, replication is policy-based—snapshots taken at specified intervals are replicated and retained for specified periods.

ActiveCluster synchronous replication updates instances of volumes in a stretched pod between two peer arrays. Administrators and hosts connected to either array have full read-write and administrative access to volumes in the pod.

Pure1 constantly monitors array states and proactively generates notifications to Pure Storage Support upon receipt of potentially problematic alerts. Its servers continuously analyze phonehome reports for fingerprints of known issues and generate support cases where warranted. When direct TSE intervention is the best way to deal with an issue, an array administrator enables Remoteassist to authorize secure login by a Pure Storage TSE.

© 2020 Pure Storage, Inc. All rights reserved.

This report is the proprietary information of Pure Storage Inc.

Pure Storage, FlashArray, Pure1, and the Pure Storage Logo are trademarks or registered trademarks of Pure Storage, Inc. in the U.S. and other countries. Other company, product, or service names may be trademarks or service marks of others.

The Pure Storage products described in this documentation are distributed under a license agreement restricting the use, copying, distribution, and decompilation/reverse engineering of the products. The Pure Storage products described in this documentation may only be used in accordance with the terms of the license agreement. No part of this documentation may be reproduced in any form by any means without prior written authorization from Pure Storage, Inc. and its licensors, if any. Pure Storage may make improvements and/or changes in the Pure Storage products and/or the programs described in this documentation at any time without notice.

THIS DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID. PURE STORAGE SHALL NOT BE LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS DOCUMENTATION. THE INFORMATION CONTAINED IN THIS DOCUMENTATION IS SUBJECT TO CHANGE WITHOUT NOTICE.