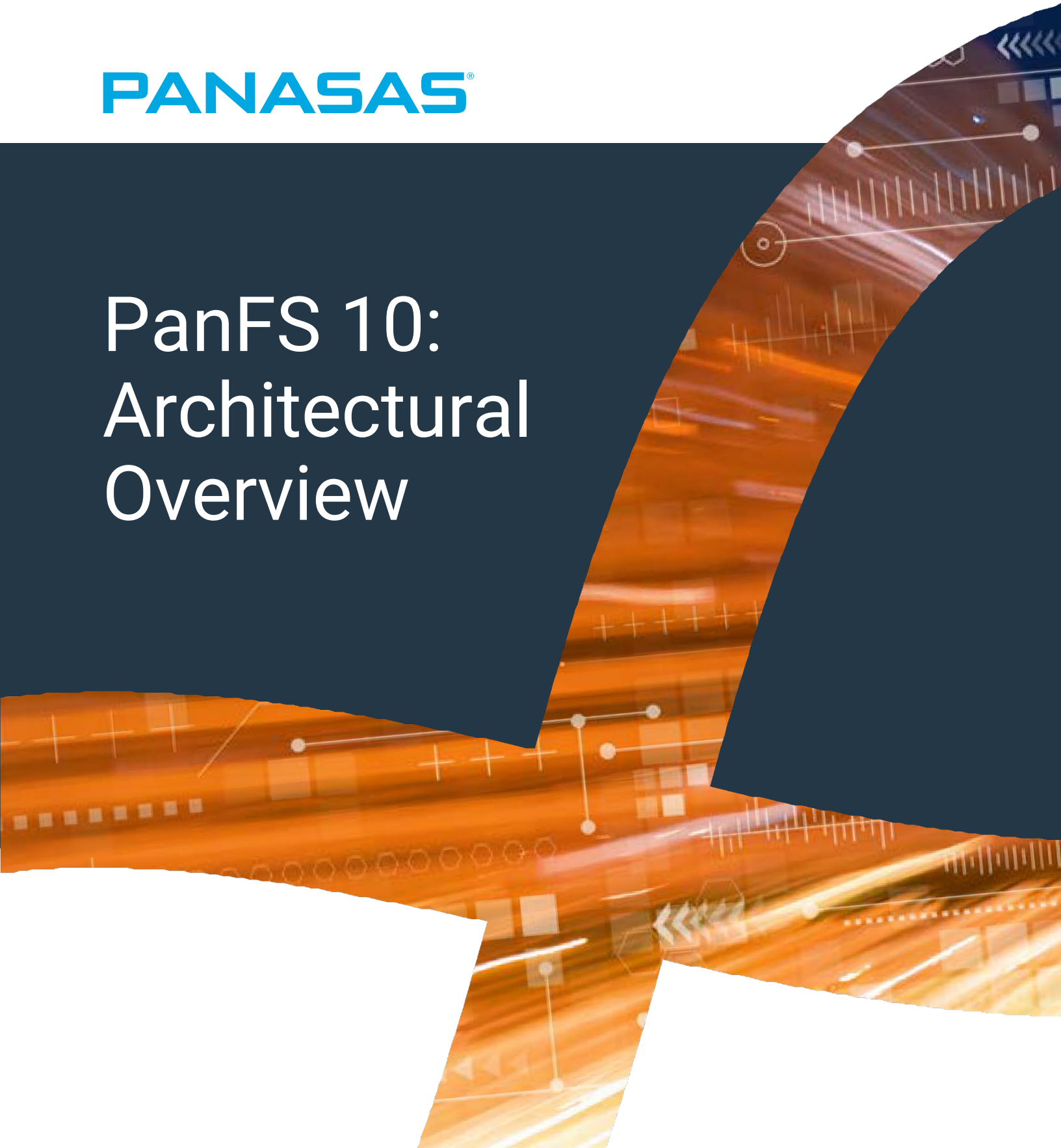
PANASAS®

PanFS 10: Architectural Overview



Contents

PANASAS

Executive Summary

High-performance computing (HPC) environments, by their very nature, tend to be large and quite complex. When it comes to pushing the boundaries in life and physical sciences or fueling the convergence of engineering innovation and artificial intelligence, it takes many computers operating together to simulate, analyze, or visualize the problems at hand. The massive quantity of data and the access performance needed to keep all those computers busy can only be achieved by a true parallel file system, one that is easy to use, resilient, and simply a fast total-performance storage system.

The Panasas® PanFS® parallel file system delivers the highest performance among competitive HPC storage systems at any capacity, and it also eliminates the complexity and unreliability associated with traditional HPC storage solutions. What's more, it accomplishes all this using commodity hardware at competitive price points.

PanFS orchestrates multiple computer systems into a single entity that serves your data to your compute cluster. Through sophisticated software, multiple systems work together to support gigabytes to terabytes per second of data being read and written by your HPC applications. PanFS manages this orchestration without manual intervention, continuously balancing the load across those systems, automatically ensuring resilience, scrubbing the stored data for the highest levels of data protection, and encrypting the stored data to safeguard it from unwanted exposure.

PanFS was the first storage system designed with the parallel file system architecture that is the de facto dominant storage architecture in HPC systems to this day. While the foundation for PanFS was laid over 20 years ago, the file system continues to leverage the latest technology advancements to provide the exceptionally high performance, unmatched reliability, and low-touch administration our customers have come to expect and rely upon.

In this document, we're going to take a breadth-first tour of the architecture of PanFS, looking at its key components then diving deep into the main benefits.

The Core of the PanFS Architecture

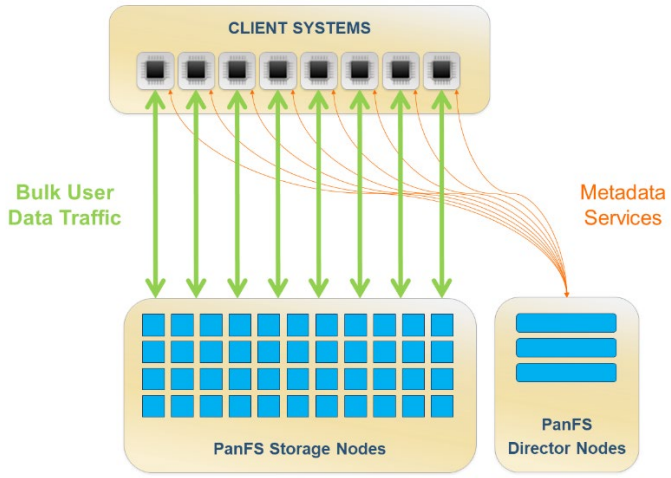


Figure 1: The PanFS Parallel File System

The Preeminent HPC Storage Architecture

There are three components working together to power the PanFS file system: director nodes, storage nodes, and the DirectFlow Client driver. The director nodes and storage nodes are computer systems dedicated to running PanFS software, and together they comprise the Panasas PanFS storage solution. The DirectFlow Client driver is a loadable software module that runs on Linux compute servers ("Clients") and interacts with the director nodes and storage nodes to read and write the files stored by PanFS. Any required administration happens via the GUI or CLI running on a director node. There's no need to interact with storage nodes or the DirectFlow Client driver – the director nodes take care of everything.

All the director nodes and storage nodes work together to provide a single file system namespace that we call a "realm." All the Linux compute servers running DirectFlow Clients that access a realm are not considered part of that realm; instead, they are considered Clients.

Separation of Control and Data Planes

PanFS explicitly separates the "control plane" from the "data plane," and director nodes in PanFS are the core of the control plane. Director nodes:

1. Cache and modify file system metadata (e.g.: directories, file attributes, access permissions, etc.).
2. Coordinate the actions of the storage nodes and the DirectFlow Client drivers for file accesses.
3. Manage the membership status of director and storage nodes within the PanFS storage cluster.
4. Control all failure recovery and data reliability operations.

Director nodes are commodity compute servers with a high-speed networking connection, significant DRAM capacity, and a persistent store for transaction logs.

Storage nodes in PanFS are the core of the data plane. They are the only part of the overall architecture that stores data or metadata. While director nodes serve and modify file system metadata, they use storage nodes to store it. Storage nodes are systems that we've chosen for their carefully balanced hardware architecture in terms of their HDD, SSD, NVMe, and DRAM capacities, strength of CPU, networking bandwidth, etc.

The DirectFlow Client driver is a loadable file system implementation for Linux systems that is installed on compute servers and used by your application programs like any other file system. It works with the director nodes and storage nodes to deliver fully POSIX-compliant and cache-coherent file system behavior, from a single namespace, across all the servers in the compute cluster. All popular Linux distributions and versions are supported.

Linear Scale-Out – Director Nodes, Storage Nodes, and Clients

PanFS scales out both director nodes and storage nodes. For more metadata processing performance, more director nodes can be added. For more capacity or more storage performance, more storage nodes can be added.

> One PanFS customer gradually added over 1,500 storage nodes and 150 director nodes to a single PanFS realm over a period of several years and saw linear growth in performance every step of the way. This was done while supporting a user community of several thousand researchers, all running their own HPC applications at the same time.

In scale-out storage systems like PanFS, there simply is no maximum performance or maximum capacity. To achieve more performance or more capacity, simply add a few more nodes. PanFS has been designed to provide linear scale-out, e.g., adding 50% more storage nodes will deliver 50% more performance in addition to 50% more capacity.

> PanFS has been tested with 30,000 client compute nodes accessing it at the same time.

Parallel and Direct Transfers with Clients

PanFS is a parallel file system that can consistently deliver orders of magnitude more bandwidth than the standard NFS or SMB/CIFS protocols. Each file stored by PanFS is individually striped across many storage nodes, allowing each component piece of a file to be read and written in parallel, increasing the performance of accessing each file.

PanFS is also a direct file system that allows each compute server to talk over the network directly to all the storage nodes. Conventional enterprise-class products will funnel file accesses through "head

nodes" running NFS or SMB/CIFS and then across a backend network to other nodes that contain the storage media that hold the file. This can create bottlenecks when data traffic piles up on the head nodes, plus it brings additional costs for a separate backend network. In contrast, for each file that the application wants to access, the DirectFlow Client on the compute server will talk over the network directly to all the storage nodes that hold that file's data. The director nodes are out-of-band, which makes for a much more efficient architecture and one that is much less prone to the hotspots, bottlenecks, and erratic performance common to traditional NAS systems.

File Maps, Parallelism, and Erasure Coding

PanFS makes use of the multiple storage nodes by assigning a map to each file which shows where all the striped component parts of that file can be found, and which storage node holds each part. The DirectFlow Client uses that map to know which storage nodes to access, directly as well as in parallel.

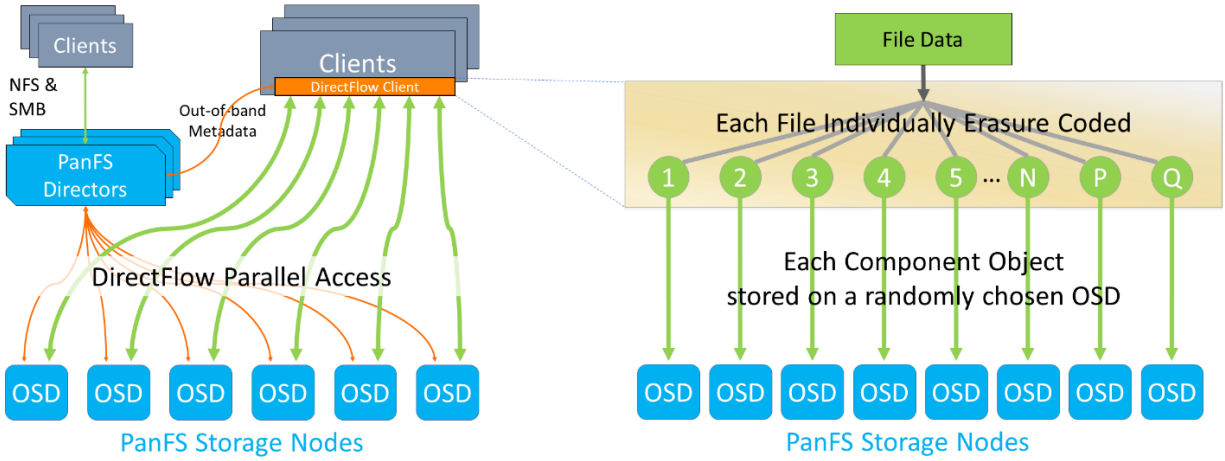


Figure 2: Per-File Erasure Coding Across OSDs

PanFS also uses Network Erasure Coding as part of that striping to ensure the highest levels of data integrity and reliability. More detail on erasure coding and "Component Objects" in PanFS can be found later in this document.

Full POSIX Semantics with Cache Coherency

The POSIX standard defines the semantics of modern file systems. It defines what a file is and what a directory is, what attributes each has, and the open, close, read, write, and lseek operations used to access files. Billions of lines of software have been written that leverage the POSIX standard for access to storage.

> All processes running on all the compute servers will see the same file system namespace, metadata, and user file data contents, all the time.

The DirectFlow Client provides the same semantics as a locally-mounted, POSIX-compliant file system. It ensures that if some other process (possibly on another compute server) is writing to a file at the same time this process is reading from it, this process will not read stale data. In file system terminology, PanFS has been engineered to provide cache coherency across all the nodes running the DirectFlow Client.

PanFS supports “access timestamps” as well as “modification timestamps” on files.

Director Nodes, the Repset, and the President

Any realm needs a minimum of three director nodes, but there’s no maximum. The administrator should designate three or five director nodes – the odd number makes it easier to break ties in voting – out of the total set of director nodes in the realm to be the rulers of the realm. This group of rulers is called the “repset” because they each have an up-to-date, fully replicated copy of the configuration database for the realm. Those rulers elect one of themselves to be the realm president who coordinates all the configuration, status, and failure recovery operations for the realm. If the director node currently designated as the president were to fail, another member of the repset would be immediately and automatically elected the new president. The configuration database is kept up to date on all members of the repset via a distributed transaction mechanism.

Director nodes make themselves available to do any of a large set of operational and maintenance tasks the president may need them to do. Those include managing a number of Volumes, being a gateway into PanFS for the NFS and/or SMB/CIFS protocols, helping perform background scrubbing of user data, helping to recover from a failure of a storage node, and helping to perform Active Capacity Balancing across the storage nodes in a BladeSet, among others. The president’s decisions can change over time as circumstances change, e.g., moving a gateway or a Volume to a different director node. However, the president will only do that if the operation is fully transparent to client systems.

How PanFS Delivers Performance

Wide Mixed Workload Performance Profile

The PanFS architecture not only delivers exceptionally high, but also very consistent, performance for workloads that include a wide range of file sizes and access patterns as well as workloads that change significantly over time.

The effect is a dramatic broadening of the use cases that PanFS can support in an HPC environment compared to other parallel file systems. All other parallel file systems require time-consuming and laborious manual tuning and retuning as workloads change.

PanFS has a wide performance profile that supports high performance when accessing a range of small files and large files. This versatility is particularly valuable in handling unpredictable workloads, such as those encountered in AI/ML projects, a shared resource center with many users running jobs on the HPC cluster, hosting home directories, and traditional HPC large file workloads.

Preventing Hot Spots

The random assignment of Component Objects to storage nodes spreads the load from file accesses across all those nodes. In most PanFS installations, the number of storage nodes is much larger than the typical stripe width of a file, so each file is very likely to only share a few storage nodes with any other files. This greatly reduces the odds of any one storage node becoming overloaded and impacting the performance of the whole realm. The result is much more consistent system performance, no matter what workload is being requested by the compute servers or how it changes over time. PanFS does this without any tuning or manual intervention.

No "hotspots" means that all the storage nodes are evenly loaded – they are all contributing equally to the performance of the realm.

Since scalable performance depends upon spreading all the files relatively evenly across the pool of storage nodes, PanFS includes Active Capacity Balancing. If the balance is off by more than a threshold – for instance, if many files are deleted at once and an OSD ends up significantly less utilized than the others – the realm president will ask the pool of directors to examine the utilization of all the storage nodes and transparently move Component Objects from over-full storage nodes to underutilized storage nodes while the realm is online.

Active Capacity Balancing happens seamlessly and continuously without requiring any administrator intervention.

Active Capacity Balancing is also used when new storage nodes are incorporated into a realm. Immediately after they are added, the realm will begin using them to store parts of any newly created files. In the background, since the utilization of those new storage nodes is so much lower than the utilization of the existing storage nodes, Active Capacity Balancing will begin moving Component Objects from the existing storage nodes to the new storage nodes. The new storage nodes will immediately start contributing to the performance of the realm and will gradually pick up more and more of the realm's workload until all storage nodes are contributing equally to the overall performance of the realm again.

PanFS can be configured to create a BladeSet for different classes of storage nodes. For example, storage nodes with a capacity of 280 TB each should not be combined into the same BladeSet as storage nodes with a capacity of 132 TB each. This helps to evenly spread the workload across the pool of storage nodes and avoid hotspots. PanFS can support multiple BladeSets in a realm and in the same namespace at the same time. Any given Volume can only draw capacity from a single BladeSet. Adding storage nodes, whether in the same BladeSet or a new one, is a non-disruptive operation performed while the file system is online.

Storage Nodes Are Sophisticated in their Own Right

In addition to the performance and reliability advantages that come from the overall PanFS architecture, there are significant performance optimizations in the PanFS storage node software that enable the most efficient use of the available storage media inside each storage node. PanFS is designed to handle combinations of up to four different performance "tiers" including Storage Class Memory such as CXL 2.0's pmem, latency optimized NVMe SSDs, capacity optimized SSDs, and HDDs.

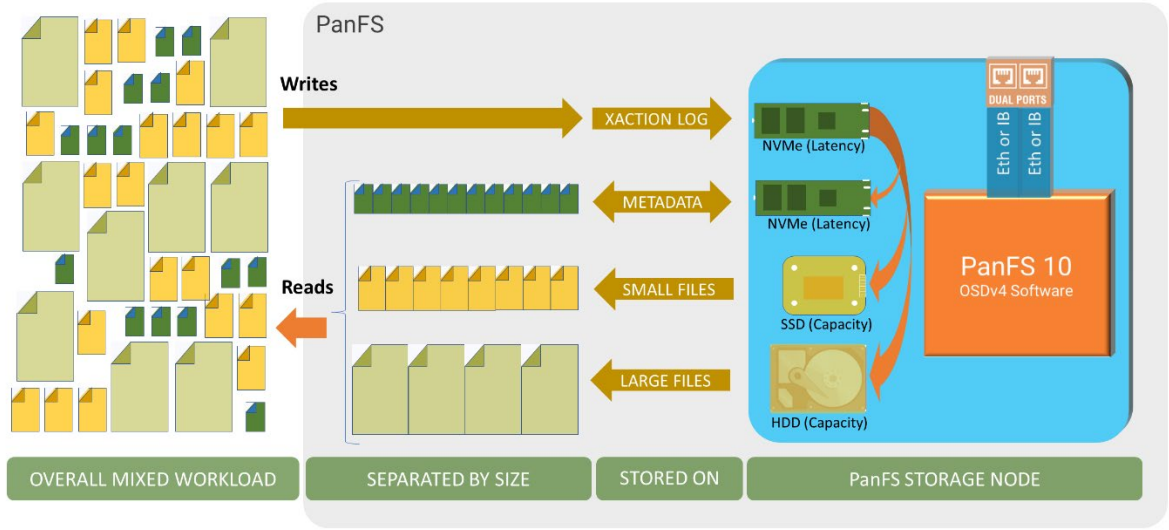


Figure 3: Data Placement Policy Inside PanFS Storage Nodes

The storage node software stack separates the storage of metadata from the storage of data. Metadata is usually composed of very small records (e.g., inodes) that are accessed in unpredictable patterns and are always latency sensitive. Directories are also metadata and are latency sensitive, but they are, more often than not, accessed sequentially (e.g., "/bin/ls"). As a result of being small, typically having unpredictable access patterns, and being latency sensitive, metadata deserves a different storage mechanism than files full of user data, which are typically much larger and accessed sequentially.

The PanFS storage node architecture results in three significant advantages: Each storage device only performs operations it is best suited for so we get more from it; small file accesses never wait for large files since they're on separate devices; and latency-sensitive metadata accesses never wait for anything.

The storage node software stack stores metadata in a database in one of the higher tiers of storage drives, typically an NVMe SSD, and stores bulk user file data in one of the lower tiers of drives, typically capacity-optimized SSDs or HDDs. It uses the highest available tier of drives as a transaction log, committing the incoming data, metadata, or operations to stable storage, therefore allowing the application to continue its processing as quickly as possible.

The storage node software stack can also support the new generations of dual-actuator HDDs that offer roughly double the bandwidth and IOPs per TB of capacity as single-actuator HDDs, resulting in performance levels that are a step function above traditional HDDs.

Consistent Mixed Workload Performance

The price/performance and mixed-workload performance of a storage subsystem depends heavily upon how effectively its architecture makes use of the performance of the underlying commodity storage devices (e.g., HDDs, SSDs, etc). Panasas uniquely excels at getting the most performance from all those devices.

As a result of focusing on using each type of storage device only for what it is best suited for, PanFS can deliver twice the performance from a given overall capacity point as other products.

PanFS takes advantage of the DRAM in each storage node as an extremely low-latency cache of the most recently read or written data and metadata. Even capacity-optimized SSDs provide cost-effective and high-bandwidth storage as a result of not having any seek times, so that's where small Component Objects are stored. Any POSIX file of less than about 1.5 MB in size will be fully stored on SSDs, with that size automatically adjusting itself up and down over time to ensure the SSDs are fully utilized.

HDDs provide high-bandwidth data storage if they are never asked to store anything small and only asked to do large sequential transfers. Therefore, only large Component Objects are stored on low-cost-per-TB HDDs.

Our storage nodes' balanced architecture gives us the optimal price-to-performance ratio. Nothing is over- provisioned, and everything is working at peak efficiency.

To gain the most benefit from an SSD's performance, we try to keep each SSD about 80% full. If one falls below that level, PanFS will, transparently and in the background, pick the smallest Component Objects from the next slowest set of drives and move them to the SSD until it is about 80% full. If the SSD is too full, PanFS will move the largest Component Objects on the SSD to the next slower tier of drives. Every storage node performs this optimization independently and continuously. It's easy for a storage node to pick which Component Objects to move; it just needs to look in its local metadata database.

The Most Performance-Efficient HPC File System

Comparing the efficiency of HPC storage systems can be difficult. The range of architectural assumptions and technologies applied by each make apples-to-apples comparisons very difficult, so we need to start by deciding on a metric of comparison. We believe that it's fair to compare hybrid configurations that contain both HDDs and SSDs to other hybrid systems and to compare all-flash deployments to all-flash deployments.

In hybrid systems, the dominant share of the total capacity in the system will be in the cost-effective HDDs, so measuring the peak aggregate performance delivered to the compute cluster divided by the number of HDDs in the storage cluster is an easy-to-understand approach. It shows how efficient the file system architecture is and how much performance it can extract from a given number of HDDs.

Single-Tier Solution Versus Complex/Costly External Tiering

Storage nodes in PanFS are typically designed using an all-hot design principle. We first decide how many of each type of storage device (e.g.: NVMe SSDs, SSDs, HDDs, etc.) will be in a new storage node. We then ensure that there is enough network bandwidth in and out of that storage node to keep every storage device in that node completely busy all the time. And finally, we include appropriate CPU power and DRAM in each node to drive both the network and the devices to full performance.

In contrast, other storage solutions may segregate their flash-based devices into a hot tier and the more cost-effective HDDs into a cold tier in an attempt to reduce the overall average cost per TB. Typically, the cold tier takes the form of a heavyweight, separately administered cold archive (e.g., with an S3 interface), plus an additional, separately administered product that will move data between the hot tier and the cold tier. Such tiered storage solutions move files between their hot and cold tiers based upon temperature – how recently a file has been accessed. That's based upon the simplistic theory that if you haven't accessed a file in a while, you won't access it for a while longer. For some important workloads such as AI, that simply isn't true.

> All the storage hardware you buy should be contributing to the performance you need.

PanFS support for multiple storage media types forms an elegant storage platform that can automatically adapt to changing file sizes and workloads, with all the underlying devices directly contributing to the application performance you need. With heavyweight tiering, the hardware in the cold tier cannot contribute to the performance the application needs, since an application can typically only directly access the hot tier. That results in three types of costs: the stranded performance costs of the drives in the cold tier not being able to contribute to application performance; the monetary costs of the additional networking and hot tier storage performance required to move data between the hot and cold tiers without impacting application I/O performance; and the direct costs of administering two separate tiers plus the policies required to move data back and forth. The cost of skilled employees is a significant piece of the overall TCO of HPC-class storage systems.

AI, Protocols, Data Management, Clouds, Edge, and Security

AI Training and AI-Powered

Training AI/ML models is an I/O intensive process at which PanFS excels. It is a read-dominated workload across a large set of input files, with only an infrequent write out of a large "checkpoint" file. AI training requires randomness in the order that the data files are accessed, and that randomness defeats the typical LRU caching of metadata. As a result, storage architectures that are heavily dependent upon caching for their performance are at a disadvantage. The focus of PanFS on separating metadata and storing it on low-latency drives offers good performance, despite that randomness.

There is also a more-direct use of AI in HPC, sometimes called "AI-Powered," where the application is *using* AI rather than the application *being* AI. For instance, the Navier-Stokes equation that is core to computational fluid dynamics is extremely costly to calculate. A neural network can provide close approximations of the values that Navier-Stokes would generate, but in a much less costly way. The approximations are used to narrow in toward an optimal result, then the real Navier-Stokes can be used for a fine-tuned final result.

PanFS was designed at the macro level of the overall architecture, and the micro level of the storage node software stack architecture, to support mixed workloads very well. It offers a wide performance profile rather than a high but narrow peak.

Data Access Protocols – DirectFlow, NFS, SMB/CIFS, and S3

As a result of its parallel and direct nature, the DirectFlow protocol will always be the highest performance path to PanFS storage, but NFS and SMB/CIFS are still useful for laptops and/or workstations to access the results of HPC jobs or for casual access to the data files stored in PanFS.

One of the roles of the director nodes in PanFS is to act as gateways that translate NFS, SMB/CIFS, and S3 operations into DirectFlow operations. PanFS provides high performance NFSv3, SMB/CIFS v3.1 (via a recent release of Samba), and S3 access to the same namespace and data files as the DirectFlow Client provides to the HPC compute cluster.

The S3 protocol support in PanFS enables cloud-native applications that use S3 as their primary data access protocol to store and retrieve data in PanFS. Objects stored in PanFS via the S3 protocol are visible as POSIX files in PanFS via the other data access protocols (DirectFlow, NFS, and SMB/CIFS), and POSIX files stored in PanFS via any other data access protocol are accessible as Objects via the S3 protocol.

Data Management – Volumes, Snapshots, and Quotas

At least one Volume must be created at the root of the PanFS file system namespace, but multiple Volumes are recommended. While each Volume is a separate unit of administrative control, they all share the physical capacity of the BladeSet (or realm if only one BladeSet exists). For example, each Volume could have a different set of per-user quota values or snapshot schedule. Each Volume is just a normal directory tree in the PanFS namespace, except that hardlinks that cross between one Volume and another are not supported.

Per-Volume snapshots are a convenient way to enable user-directed recovery of prior versions of files, with no system administrator involvement required. They are accessed via the typical hidden and synthetic ".snapshots" subdirectory in each directory in the namespace.

Both soft and hard quotas are supported, at both the user level and Volume level. Each user can have their own soft and hard capacity quota for each Volume, and each Volume can be configured with its own soft and hard quota for total capacity consumed by all users in that Volume.

Data Security – ACLs, SELinux, and Encryption at Rest

PanFS supports two features that prevent unauthorized data access while the realm is online – ACLs and SELinux, and one that prevents unauthorized data access while the realm is offline – Encryption at Rest.

PanFS supports Access Control Lists (ACLs) on each file and directory in addition to the traditional Linux user ID, group ID, and mode bits such as "joe dev -rwxr-xr-x." PanFS ACLs are fully compatible with Windows ACLs and ActiveDirectory-based and LDAP-based account management, and provide fine-grained control over which user accounts can execute which operations on each file or directory.

> Data security is becoming an ever-more important characteristic of storage systems, and Panasas will continue developing solutions that protect your data from unauthorized access.

Running SELinux on the client systems that access PanFS via DirectFlow provides a kernel-implemented set of mandatary access controls that confine user programs and system services to the minimum level of file and data access that they require. PanFS integrates the security.selinux security label that enables this into the PanFS inode structure, resulting in near-zero added access latency when SELinux is used. This is the foundation for true Multi-Level Security policy implementations that allow

users and data in different security levels and compartments to share the same compute and storage resources.

All Panasas storage nodes use industry-standard self-encrypting drives (SEDs) which implement NIST-approved AES-256 hardware-based encryption algorithms built into each drive. These drives are specifically designed so that encryption does not reduce performance. PanFS allows encryption-at-rest to be non-disruptively enabled and disabled, drive keys to be rotated upon command, and cryptographic erasure for securely repurposing the storage without risking exposure of the prior data.

Key management is outsourced via the Key Management Interoperability Protocol (KMIP) to well-established and proven cyber security key management solutions that provide centralized and simplified key management. During normal runtime, PanFS periodically verifies that the KMIP server is alive and well and contains all the right keys, and it will raise an alert if there is any type of problem.

PanView – Analytics Insights into the Population of Files

PanFS includes a highly optimized API called PanView for bulk retrieval of the POSIX metadata of files stored in PanFS and supports that metadata being exported to visualization tools for analysis. An optional advanced version of PanView enables full graphical analytics reports and ad-hoc queries without the need for third-party tools, especially web-based tools on the public Internet.

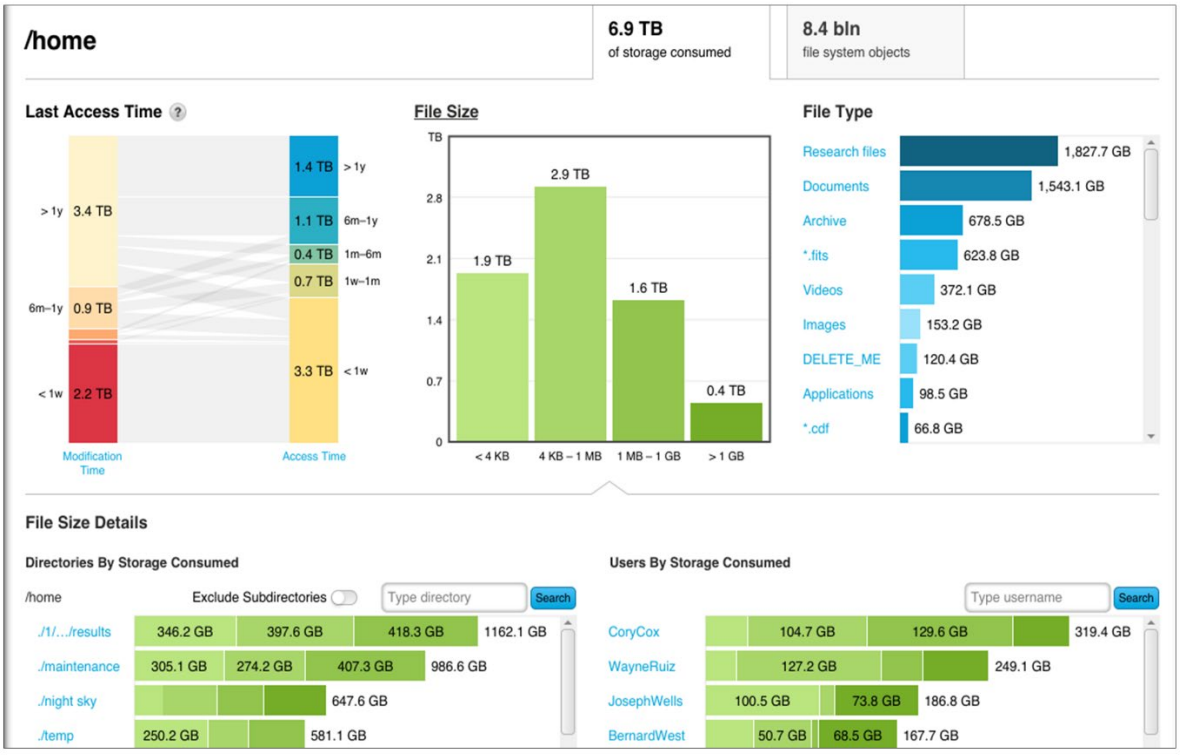


Figure 4: PanView Analytics Example Dashboard

The target use cases for this capability are in identifying patterns in the population of files stored in PanFS that are actionable by the System Administrator. Use case examples include identifying which user is consuming more than their share of the capacity of a PanFS realm or identifying “reference

datasets" – groups of files that are frequently accessed (have recent access timestamps) but have not been modified in a long time (have old modification timestamps). For example, a "written three years ago" modification timestamp might typically imply that a file is cold and should be moved, even though ones that are frequently accessed are actually hot.

To enable this latter insight, PanFS supports "access timestamps" on every file that are updated at most once per day rather than once per read or write operation. This granularity is sufficient for the above use cases without imposing a performance burden on PanFS.

PanMove – Optimized Bulk Data Movement at Scale, Including To/From Clouds

PanFS includes a highly optimized and scalable data mover feature called PanMove that can copy bulk user data files to/from other systems, both PanFS realms or non-PanFS storage solutions, whether local or geographically remote. PanMove can synchronize the data between source and destination; it can also move the data by deleting the source after the copy has been made and verified. Additionally, PanView can be configured to directly trigger PanMove operations, not just inform the user that they should move some files.

The base level of this feature is an optimized and parallel "rsync" implementation that supports movement between POSIX filesystems. Since cloud storage can be used as the basis for computation and/or as archive storage for precious data, an optional advanced version of PanMove is a fully-featured parallel data mover architecture that can transfer files to/from all the major public clouds (AWS, Azure, and Google) via their native Object APIs, as well as between POSIX filesystems. PanMove can also support backup and archive operations to local media such as tape as well as cloud archive storage.

HPC in the Cloud – Hybrid On-Prem and Cloud Computing

PanFS enables a hybrid model of using an on-prem HPC compute cluster plus leveraging compute resources in the public clouds, whether that is the latest AI/ML accelerators such as GPUs or simply on-demand scalable compute resources.

For applications that are best run in the cloud, the PanMove feature of PanFS can easily, quickly, and efficiently move any number of data files between on-prem HPC deployments and AWS, Azure, or Google plus the second tier of cloud providers that support the S3 protocol. Locally acquired data files can be moved up for processing and results files can be moved back for further processing or storage.

For cloud-native applications that use S3 as their primary data access protocol and that can also be run in a local HPC cluster, the S3 data access protocol support in PanFS enables them to use PanFS for their data storage needs.

Panasas does not yet support deployment of PanFS in the cloud, but we're researching our options on how to best implement it.

HPC at the Edge – Small PanFS Deployments and Satellite HPC Centers

PanFS can be easily deployed on smaller storage clusters while still offering the core value of a parallel filesystem, whether at smaller HPC shops or satellite HPC sites that cooperate with a larger core HPC site. In the former case, the typical life sciences customer doesn't have the same storage capacity needs or compute cluster needs as a CFD customer doing aeronautical simulations or weather projections, so are sometimes classed as being at the "edge." In the latter case, satellite HPC sites may need to process locally acquired data before partially and/or fully processed data files are moved to the core HPC site, or centrally processed results files may need to be moved to the satellite HPC site for use in local processing. For example, AI models being trained at the central HPC site can be copied to satellite HPC sites with PanMove for deployment for inferencing on locally acquired data.

The Most Reliable HPC-Class Storage

Director Software and Deeply Automated Failure Recovery

In addition to being responsible for the POSIX semantics and cache coherency of the files in a Volume, director nodes also need to manage the status and health of each of the storage and director nodes that are part of the realm. Panasas has analyzed the failure modes of each of the commodity platforms that PanFS has been ported to, and we have included recovery logic for each of those cases into the director node software stack. That additional engineering work is a significant contributor to the overall reliability of a PanFS realm and is one of the keys to its low-touch administration. PanFS automatically reacts to failures and recovers from them, taking care of itself.

> Panasas has customers who have had zero unplanned downtime over the multi-year life of their PanFS realms.

Each File Is Individually Erasure Coded for Maximum Reliability

Storage nodes in PanFS are actually highly sophisticated Object Storage Devices (OSDs), and we gain the same scale-out and shared-nothing architectural benefits from our OSDs as any Object Store would. The definition of an Object used in our OSDs comes from the Small Computer System Interface (SCSI) standard definition of Objects rather than the Amazon S3 Object definition.

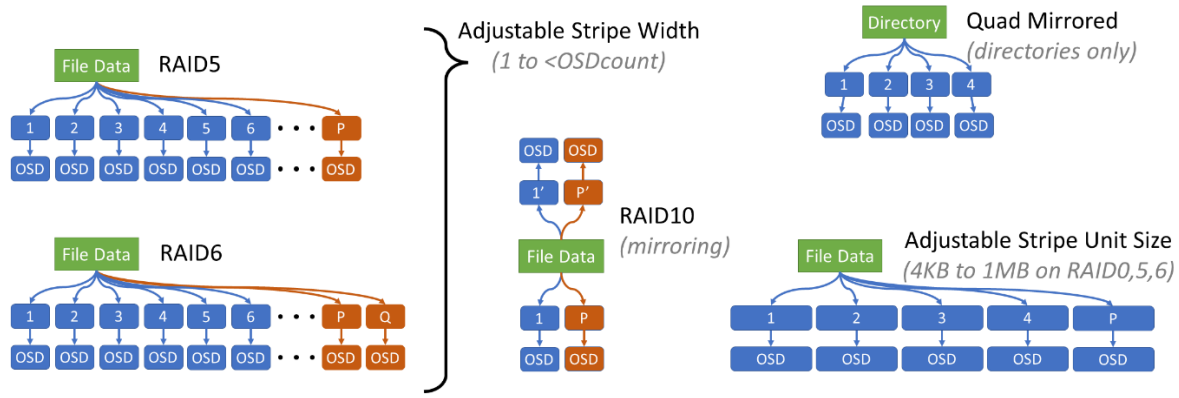


Figure 5: Per-File Erasure Coding Layouts

PanFS uses SCSI Objects to store POSIX files, but it does so differently than how S3 Objects are typically used to store files. Instead of storing each file in an Object like S3 does, PanFS stripes a large POSIX file across a set of Component Objects and adds additional Component Objects into that stripe that store the P and Q data protection values of an N+2 erasure coding scheme. Using multiple Objects per POSIX file enables the striping that is one of the sources of a parallel file system's performance.

> A RAID array reconstructs the contents of drives, while PanFS reconstructs the contents of files.

While large POSIX files are stored using erasure coding across multiple Component Objects, small POSIX files use triple-replication across three Component Objects. This approach delivers higher performance than can be achieved by using erasure coding on such small files, and makes it more space efficient, as well. Unless the first write to a file is a large one, it will start as a small file. If a small file grows into a large file, the director node will transparently transition the file to the erasure coded format at the point that the erasure coded format becomes more efficient.

When a file is created, and as it grows into a large file, the director node that is managing those operations will randomly assign each of the individual Component Objects that make up that file to different storage nodes. No two Component Objects for any file will be in the same failure domain.

Reliability That Increases with Scale

Any system can experience failures, and as systems grow larger, their increasing complexity typically leads to lower overall reliability. For example, in an old-school RAID subsystem, since the odds of any given HDD failing are roughly the same during the current hour as they were during the prior hour, more time in degraded mode equals higher odds of another HDD failing while the RAID subsystem is still in degraded mode. If enough HDDs were to be in a failed state at the same time there would be data loss, so recovering back to full data protection levels as quickly as possible becomes the key aspect of any resiliency plan.

If a Panasas storage node were to fail, PanFS needs to reconstruct only those Component Objects that were on the failed storage node, not the entire raw capacity of the storage node like a RAID array would. PanFS would read the Component Objects for each affected file from all the other storage nodes and use each file's erasure code to reconstruct the Component Objects that were on the failed node.

> PanFS has linear scale-out reconstruction performance that dramatically reduces recovery times in the event of a storage node failure, so PanFS reliability goes up with scale.

When a BladeSet in PanFS is first set up, it sets aside a configurable amount of spare space on all the storage nodes in that BladeSet to hold the output from file reconstructions. When PanFS reconstructs a missing Component Object, it writes it to the spare space on a randomly chosen storage node in the same BladeSet. As a result, during a reconstruction, PanFS uses the combined write bandwidth of all the storage nodes in that BladeSet. The increased reconstruction bandwidth results in reducing the total time to reconstruct the affected files, which reduces the odds of an additional failure during that time, which increases the overall reliability of the realm.

> Data scrubbing is a hallmark of enterprise-class storage systems and is only found in one HPC class storage system, PanFS.

PanFS also continuously scrubs the data integrity of the system in the background by slowly reading through all the files in the system, validating that the erasure codes for each file match the data in that file.

An Architecture of Even More Reliability

The N+2 erasure coding that PanFS implements protects against two simultaneous failures within any given BladeSet without any data loss. More than two failures in a realm can be automatically and transparently recovered from, as long as there are no more than two failed storage nodes at any one time in a BladeSet.

If, in extreme circumstances, three storage nodes in a single BladeSet were to fail at the same time, PanFS has one additional line of defense that would limit the effects of that failure. All directories in PanFS are stored quad-replicated – four complete copies of each directory, no two copies on the same storage node – rather than the triple-replicated or erasure coded formats used for regular files.

If a third storage node were to fail in a BladeSet while two others were being reconstructed, that BladeSet would immediately transition to read-only state, as a result. Only the files in the BladeSet that had Component Objects on all three of the failed storage nodes would have lost data, which becomes a smaller and smaller percentage as the size of the BladeSet increases. All other files in the BladeSet would be unaffected or recoverable using their erasure coding.

Compared to all-or-nothing architectures, PanFS has a much more intelligent, graceful failure model that we call “Extended File System Availability.”

Since PanFS would have one complete directory tree still available to it, it can identify the full pathnames of precisely which files need to be restored from a backup or reacquired from their original source, and can therefore also recognize which files were either unaffected or recovered using their erasure coding.

PanFS is unique in the way it provides clear knowledge of the impact of a given event, as opposed to other architectures which leave you with significant uncertainty about the extent of the data loss.

We like to say that extraordinary architecture leads to extraordinary products.

Conclusion

PanFS has deep roots in the HPC business. Panasas has developed and contributed many core aspects of the architectures and best practices of HPC storage over the years. The focus of PanFS on reliability and low-touch administration is unique in an HPC-class storage system and bridges the gap between traditional HPC that focused on performance to the detriment of reliability and the new reality of HPC moving into the enterprise as a core service.

The term “scratch storage” that has traditionally been applied to HPC-class storage systems implies two core things: that the storage system is fast, but that it is unreliable. With PanFS, you no longer must choose – PanFS is both very fast and very reliable. It can store and serve the high-performance intermediate files of HPC compute jobs, while at the same time storing and serving all the home directories and ancillary files of your user community, and it can do that with enterprise-class stability, data reliability features, and ease of administration.

PanFS is the first HPC-class storage system that can meet all the needs of a high-performance environment.

11/2023

PANASAS®

2680 N. First St., Suite 150
San Jose, CA 95134

PH +1.888.PANASAS
F +1.408.215.6801
www.panasas.com