WHITE PAPER

The Metadata Advantage

Scaling I/O performance for AI and HPC
with Pure Storage FlashBlade//EXA™

Contents

Abstract

It's not overstating to assert that metadata, sometimes called "data about data," is a vital part of information technology. Every data item (file or object) held in a storage system is associated with metadata that identifies and describes it—its name, location, size, ownership, creation time, and so forth. *Clients* (users of data), both humans and computer applications, access metadata to read, write, and manage data item contents; *data managers* (file systems and object stores) validate permissions and provide the information required to gain access.

Historically, metadata has been stored with the data item contents it describes, for example in file system directories. As the amount of data that users keep online has grown, especially with *high-performance computing* (HPC) and *artificial intelligence* (AI) applications, metadata access has emerged as a significant bottleneck. The bottleneck is alleviated by parallel storage systems that use a metadata service to manage data items whose contents are stored in dozens, or even hundreds, of separate data servers, typically accessed via a separate network.

This white paper describes the Pure Storage® **FlashBlade//EXA™** *parallel file system*, whose *core metadata cluster* can provide thousands of concurrent clients with fast access to metadata that manages billions of data items stored in hundreds of separate data servers. With I/O performance and storage capacity that scale well beyond those of conventional file servers, **FlashBlade//EXA** systems are a perfect centerpiece for multi-petabyte AI and HPC deployments, today and in the future.

The data explosion

Ever since data was first stored digitally, users have recognized a need to organize and track items systematically, giving rise to file systems, and more recently, to object stores. Both use *metadata*—names, sizes, locations, access permissions, and so forth—to organize and manage stored data items. Metadata is indispensable across the information technology spectrum, from wearable appliances to supercomputers. In large storage systems that contain billions of items, consistent fast access to metadata is a necessity for application viability.

As storage system capacities increase, systems tend to serve more concurrent clients. Client access to metadata, usually co-located with content, competes with content transfers for storage system and network bandwidth. In *high-performance computing* (HPC) applications that analyze experimental results, conduct simulations, and so forth, dozens of client computers may read and/or write tens of thousands of large (multi-megabyte) data items concurrently. HPC's I/O throughput needs, well beyond the capability of any one storage system, become a bottleneck for clients' attempts to access items via their metadata.

Containing the data explosion

Storage developers responded to HPC's storage and I/O needs with *parallel* (sometimes called *clustered*) storage systems[1], the best-known of which are IBM Corporation's *General Parallel File System* (GPFS[2]), and the *Lustre* open-source software package[3].

Implementation details vary, but parallel storage systems implement some variety of the model shown in Figure 1. They *disaggregate* (separate) metadata and data content and ideally provide clients with separate paths for accessing each. Some use a single metadata server, others a *metadata cluster*. These systems usually employ specialized client-side software and most require highly skilled support teams.

With this model, metadata and data can scale independently. For applications that mainly process very large data items, data servers and network bandwidth can be added to increase data capacity and/or data transfer performance as needed without forcing users to add metadata capacity that they don't need. Conversely, applications like image repositories that manage millions of items of modest size can expand metadata as needed without being forced to configure additional content capacity for which they have no use.

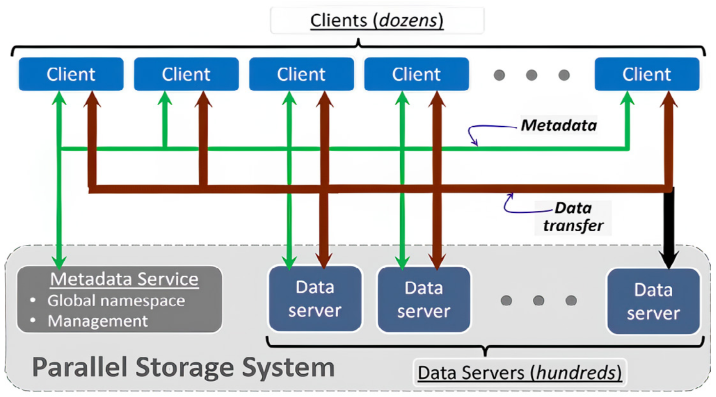

FIGURE 1 Parallel storage system model

Assuming a balanced I/O load, a parallel storage system's maximum throughput is determined by the bandwidth potential of the data transfer network and the aggregate data transfer capability of its data servers. Client access to metadata is not in the data transfer path.

Parallel storage systems are a mainstay of HPC. In 2010, the *Internet Engineering Task Force* (IETF) ratified the *Network File System* (NFS) version 4.1 standard, which includes a *parallel NFS* (pNFS) client access protocol specifically for parallel *storage* systems. The pNFS protocol defines a central metadata service and allows for file, object, or block-based data servers, all of which present data items to client applications as files in the overarching pNFS file system.

The single namespace

A parallel storage system's metadata service presents a single *namespace* (set of all possible data item names) to clients. Its metadata service *maps* (relates) the names of stored items to the locations of their content and balances the system's capacity utilization and I/O load.

In parallel file systems, for example, each data server presents a separate abstraction (usually a local file system) to clients. System metadata maps names in the namespace to data server-specific local identifiers (file handles). Client-side software specifies these identifiers when transferring item contents directly to and from data servers.

Enter artificial intelligence

Parallel storage systems are widely accepted in HPC, so when AI began to gain momentum, they were an obvious choice for the enormous repositories that AI deployments create and use for data preparation, training, and inferencing. Like HPC, AI needs enormous amounts of data transfer bandwidth, but it presents even greater challenges for parallel storage systems:

More clients

In a distributed HPC application, dozens of client computers may access the same storage system concurrently. AI models, however, are typically trained by clusters of hundreds or more computing servers, each containing an array of graphics processing units (GPUs) and a control computer that interacts with storage and other parts of the environment. For full utilization, GPUs must be provided with constant streams of data, so even preliminary training experiments can require terabytes per second of data transfer.

More experimentation

Whereas most HPC applications use mature algorithms to process data, in the early phase of AI model development *data* scientists may conduct dozens of preliminary experiments, each accessing a subset of curated data items as input. Typically, many of these jobs run concurrently, creating heavy aggregate load on a system's metadata infrastructure as they make frequent requests for access to data items.

More data items

AI storage systems typically manage many times more data items than HPC deployments. Large deployments create billions of *curated* items by transforming raw input data in preparation for model training. Training jobs keep GPUs occupied by retrieving millions of curated items per second via their metadata.

More complex usage patterns

Whereas HPC applications predominantly process multi-megabyte data items, concurrent access to data items in a typical AI deployment can range from gigabyte-size video clips and similar inputs, to megabyte-size frame captures and documents, to thumbnails, sensor readings, and text snippets in the kilobyte range.

- It is not unusual for data preparation to create billions of relatively small data items by transforming (randomizing, filtering, resampling, etc.) raw input data. Storage systems generate metadata for every item created.
- Training jobs access metadata to locate and read curated items but they also generate metadata as they write gigabyte-size checkpoints as often as every few minutes.
- Some production models process ad hoc queries that are accompanied by (possibly voluminous) supporting collateral; others receive constant inputs from automated processes or event recordings. Whatever the source of demand, models constantly intermix input requests with frequent access to model elements as they produce output *inferences*.

AI storage systems must excel at all these usage modes, often simultaneously as they support concurrent data preparation, training, and production jobs. In short, systems must be able to handle rapidly (and unpredictably) changing read/write mixes of data items in a wide range of sizes requested by thousands of clients.

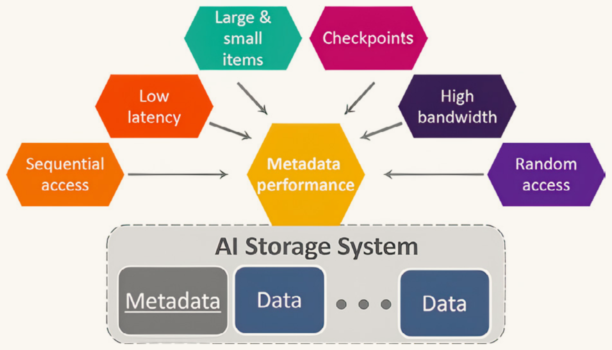

FIGURE 2 Demands on AI storage

While AI throughput demands equal or exceed those of HPC, the technology presents a more complex challenge for data storage with requirements for fast, reliable response to rapidly changing workloads as applications running on thousands of clients produce and consume billions of data items that vary widely in size, with each access requiring creation of or access to existing metadata.

“Tuning” AI storage for well-defined workload properties doesn’t work.

Using metadata at scale

In HPC and AI deployments, it is common for many users to access overlapping sets of data items concurrently. Dozens of HPC clients analyze subsets of experimental data from different perspectives. AI model training requires a constant flow of curated input to each of potentially thousands of GPUs. In both cases, the result is a constant stream of metadata demands of two types:

Access to individual data items

The most obvious uses of metadata are by clients accessing data items to read or write their content, and creating new items, each of which requires metadata to be generated. Clients of parallel storage systems request access from the metadata service; the metadata service validates requests and responds with locations of item contents in one or more data servers. During AI training, a metadata service may receive millions of metadata requests for read access every second. Data preparation applications may create thousands of new items (and their metadata) every second.

Bulk operations

Client requests to enumerate (list) data items, change item access modes or ownership, move items between directories, and remove (delete) items typically specify hundreds or more data items as targets. For the most part, they do not access item contents, but they do read and usually update the metadata of every targeted item.

In many common scenarios, these requests can overwhelm storage system metadata services. For example, a parallel file system client's CHMOD or CHOWN command that targets a directory containing a million files can run for hours as each change is made individually, preventing downstream HPC analysis or AI training jobs from starting. Slow bulk operation performance is a shortcoming of many parallel file systems.

While today's AI and HPC parallel storage systems have a primary goal of delivering high throughput in multi-petabyte deployments, their success depends equally on how efficiently they process metadata requests intermixed with data transfer.

FlashBlade//EXA

A **FlashBlade//EXA** system consists of a metadata cluster delivered by Pure Storage, integrated with a number of data nodes[4] (commodity servers). Figure 3 illustrates the system's basic architecture.

The system's key feature is the use of separate devices for holding data and metadata and separate network paths for accessing them. Clients use the **metadata path** to request access to existing data items and to create new ones. The metadata cluster uses it to allocate data node space for new items and to control the data nodes. All content is transferred directly between clients and data servers on the **data transfer path**.

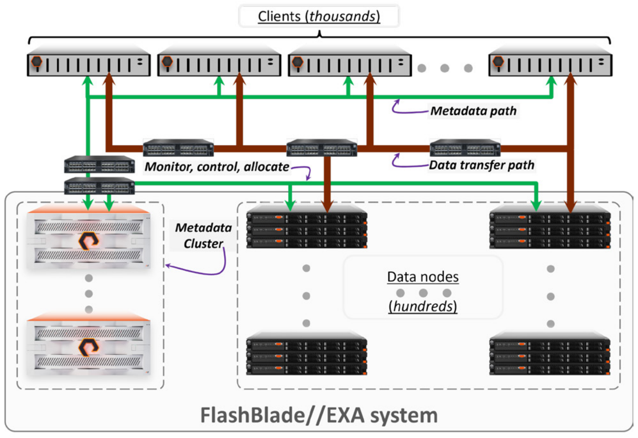

FIGURE 3 FlashBlade//EXA architecture

Hardware

Metadata cluster hardware consists of one or more *chassis*[5] built from FlashBlade//S™ technology. FlashBlade//S is the highest-performing member of the FlashBlade® family, field-proven over eight years with thousands of production deployments. Each chassis holds between seven and 10 blades that contain processing, buffer, and communication resources and mounting slots for up to four *DirectFlash™ Modules* (DFMs). Each DFM contains flash memory for persistent storage and NVRAM for temporary *staging* of data written by clients. Blades are peers—each one has access to all system flash and NVRAM. The system's Purity software balances internal processing and I/O among blades independently of client request pattern.

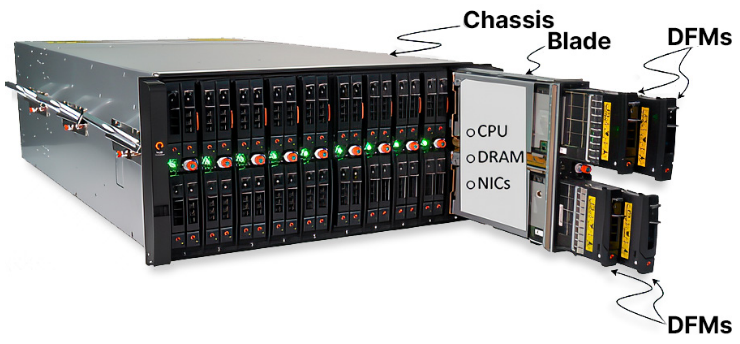

FIGURE 4 FlashBlade//S chassis

Users can deploy any type of data node that meets some basic requirements for CPU type, DRAM size, number and capacity of storage devices, and Ethernet interfaces. Data nodes should be capable of 100GB/s of read throughput and about half that for writing. Pure Storage anticipates that some users will redeploy pre-existing server "farms" as data nodes.

Metadata cluster software

FlashBlade//EXA initially implements the pNFS file system standard for client access.

Metadata clusters run the same Purity software as other FlashBlade family members, augmented by a pNFS server for interaction with clients.[6] Figure 5 shows the principal software components:

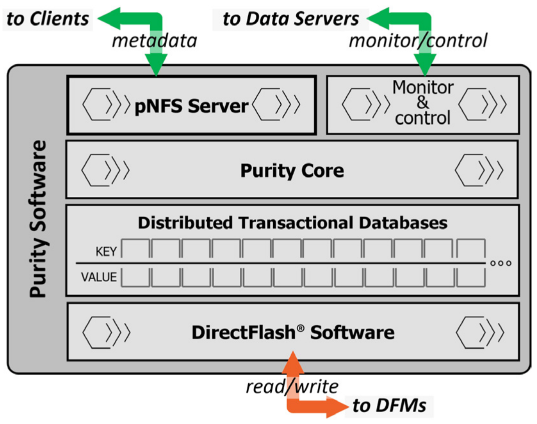

FIGURE 5 Purity software

pNFS Server

Uses the pNFS server-side file access protocol to interact with clients.

Purity core

Schedules, balances load, and manages network traffic with clients and data nodes.

Distributed transactional databases

Organizes system flash and NVRAM as a set of distributed key-value databases. The databases are an important factor in the system's high metadata performance.

DirectFlash software

Manages DFMs and cooperates with DFM firmware to read and write flash and NVRAM with almost none of the internal data movement and write amplification common with off-the-shelf flash SSDs.

Monitor and control

Monitors data node "health" and utilization, allocates storage for items created by clients, balances data node utilization, and generates alerts as needed.

FlashBlade//EXA systems use Purity's distributed key-value databases to organize the pNFS metadata of data item content stored on data nodes. Keys are names in the single namespace; values are properties—location, size, access permissions, and so forth—of stored data items.

Data node software

As part of **FlashBlade//EXA** software, Pure Storage supplies a suite of data node software that includes an executable system image, a zero-touch network-bootable installer, monitoring and visualization tools, and tools for configuration and on-going management tasks.

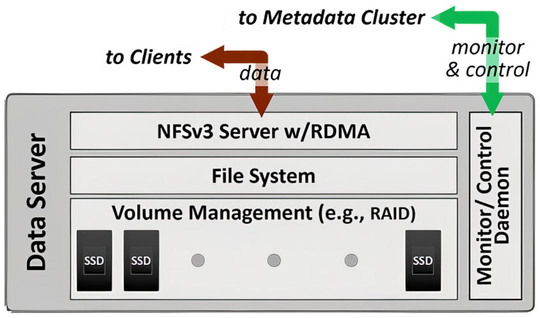

FIGURE 6 Data server

Data node SSDs are user-configurable for mirroring (for best performance) or parity RAID (for maximum capacity). Each data node implements a local file system, with which client pNFS modules read and write data via NFSv3 (using *Remote Direct Memory Access* [RDMA] to minimize data copying) and which the metadata cluster uses to allocate space for new items. The metadata cluster interacts with data node daemons to monitor node "health" and utilization and to control their operation.

System workflow

FlashBlade//EXA implements the FlexFiles[7] version of the pNFS protocol in which a file's content may be located on a single data node or striped across multiple nodes for performance.

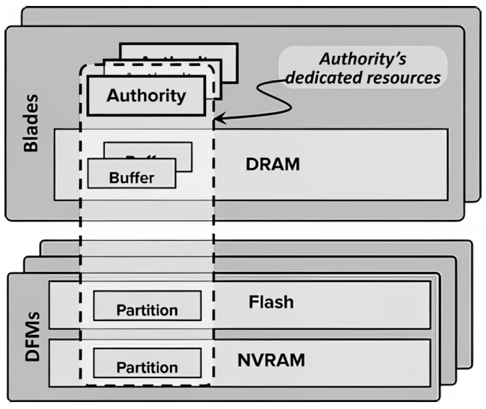

FIGURE 7 Authority resources

To access existing data items, clients issue NFSv4.2 GETATTR, STAT, OPEN, and similar commands to the metadata cluster. The cluster validates client permissions and responds with the data node locations of item contents. Client-side pNFS software transfers contents directly to and from data nodes' local file systems. The metadata cluster is not in the data transfer path.

To create new items, clients issue NFSv4.2 CREATE commands to the metadata cluster. After validation, Purity selects a data node (based on utilization that it monitors), creates an empty NFSv3 file on it, and returns the file's *layout* metadata to the client. The client writes the file's content directly to the data node and updates the metadata cluster's pNFS metadata for the item.

Purity internal workflow

From an internal Purity perspective, the **FlashBlade//EXA** pNFS metadata service is a file system. As with all FlashBlade systems, the software directs client requests to *authority* processes running on blades for execution based on a calculated hash of the requested file identifier (e.g., {pathname,offset}). Hashing ensures an even distribution of internal system load regardless of client access pattern.

Each authority manages separate *partitions* of NVRAM and flash on each of a cluster's DFMs. Authorities stage data received from clients in their NVRAM partitions to protect against power loss and system failures and place it in large DRAM buffers in compressed form for writing to flash. As buffers fill, authorities split them into *shards*, add two or more shards that contain calculated erasure codes to protect against read failure, and write each shard to a flash partition on a different DFM for resiliency.

The buffers that **FlashBlade//EXA** authorities write to flash are effectively arrival time-ordered logs of "data" (i.e., metadata updates) written by clients. Authorities do not overwrite stored data "in-place." They track each stored item's current content in a persistent structure called a *pyramid*, part of which they cache for rapid access. Background tasks *garbage collect* (required with any log structure) to remove obsolete content and reclaim the space it occupies.

A word about networking

Metadata clusters include two of Pure Storage's eXternal Fabric Modules (XFMs) for redundant inter-chassis communication and for connecting to users' **metadata** networks for client access to metadata. **FlashBlade//EXA** systems do not include data transfer network facilities for direct client-data node communication.

Why FlashBlade//EXA excels at scale

Parallel storage systems maximize throughput by separating metadata access from the data transfer path. The key to the **FlashBlade//EXA** parallel file system's outstanding metadata performance is the Purity software massively distributed scale-out design that:

- Distributes internal load evenly across a system's blades regardless of client I/O pattern.
- Balances internal I/O by assigning primary responsibility for accessing and managing disjoint segments of each DFM's NVRAM and flash memory to authority processes that run on each blade.
- Minimizes inter-blade interactions (e.g., locking)—a major performance constraint in many scale-out systems—by assigning primary responsibility for managing a disjoint segment of the single namespace to each authority process.
- Compresses data and stores it in log structures that use storage efficiently and do not cause write amplification when clients partially overwrite items. Authorities run low-priority background tasks to perform garbage collection and reorganize stored data.
- Interacts with DirectFlash software to minimize bottlenecks at the device-level and to transfer data between blade buffers and DFMs with almost zero write amplification.

Access to individual data items

Purity's pNFS service directs client OPEN and CREATE requests to authorities based on hashes of requested files' pathnames. Authorities validate requests and return layout metadata that indicates the locations of file contents. Clients transfer data directly to and from data nodes.

For CREATE requests, authorities select one or more data nodes with low utilization and make NFSv3 CREATE requests to them for zero-length items. When the data nodes respond, the authorities create NFSv4.2 metadata for the item and return its layout to the client. The client writes the item's content directly to the data nodes.

The keys to Purity's metadata performance are the load balancing design that keeps all internal resources equally busy and the relative autonomy of authorities that minimizes the need for inter-blade crosstalk (sometimes referred to as 'east-west traffic').[8]

Bulk operations

FlashBlade//EXA metadata performance is a major advantage in bulk metadata-only operations such as enumerating directory contents, changing file ownership or access mode, moving files between directories, and deleting files. The system's highly distributed scale-out design can perform millions of concurrent metadata-only operations per second. Rapid-fire series of metadata commands don't bog down waiting for previous or unrelated operations to complete. Bulk operations don't overwhelm **FlashBlade//EXA** metadata clusters.

Why FlashBlade//EXA?

Because parallel storage systems are critical to AI and HPC deployments, performance, flexibility, security, and cost are important storage selection criteria. Among storage vendors vying for user attention, this is what makes **FlashBlade//EXA** stand out from the crowd:

Robustness

Metadata is at the core of a scale-out disaggregated storage system. A failed data node impedes operation; failure of the metadata service stops it cold.

Pure Storage's decade-long track record of very high availability over tens of thousands of field deployments makes the FlashBlade//S platform a solid foundation for **FlashBlade//EXA**.

Flexibility

FlashBlade//EXA users can configure metadata clusters that meet their current needs and expand them non-disruptively by adding blades and/or chassis as their needs for metadata increase.

Similarly, they can deploy data nodes and network facilities within current performance-capacity-cost-availability envelopes and add to them non-disruptively as they accumulate additional data and/or require higher throughput.

Metadata access performance

The FlashBlade//S metadata cluster foundation is unique among scale-out system designs in minimizing internal contention. Performance scales linearly with capacity. Among users, FlashBlade systems are well-known for their high performance in metadata operations.

Data transfer performance

Users control a **FlashBlade//EXA** system's maximum data transfer throughput by adjusting its data node count and network infrastructure to meet their performance requirements.

Pure Storage culture

Since its inception, Pure Storage has striven to make storage simple, and parallel storage systems continue that trajectory. From Purity's high metadata performance, to system configuration flexibility, to world-class consulting and support organizations, to the company's verified Net Promoter Score of 81, Pure Storage makes designing, configuring, deploying, and managing **FlashBlade//EXA** systems as friction-free as it can possibly be.

Looking ahead

Throughout Pure Storage's history, from its introduction of affordable flash storage for the enterprise, to FlashBlade scale-out file and object systems, to the Evergreen® deployment and service models, to fleet-wide management with Pure1® and Pure Fusion™, to the Purity software platform, the company has consistently identified storage users' unmet needs and satisfied them.

Its style has been timely introduction of core solutions to significant problems, followed by steady enhancements that make the solutions increasingly comprehensive and easy to use.

And so it is with parallel storage systems. It's clear that today, HPC and AI's most immediate storage and I/O needs are high data throughput and reliable low latency metadata access. But both technologies are evolving rapidly, and will need innovations in performance, functionality, and ease of deployment and use, which Pure is well-positioned to deliver. For example:

- To support the larger AI training jobs of the future, **FlashBlade//EXA** metadata clusters expand non-disruptively to accommodate more data items whose content is stored in larger numbers of data nodes.
- FlashBlade's mature object storage support is a solid foundation for applications based on objects (rather than files) as they become more prevalent.
- Both FlashBlade and FlashArray™ systems improve data transfer performance and protect against device read failures by appending erasure codes to data blocks and striping them across multiple devices. The underlying erasure coding technique is an obvious mechanism for improving data node resilience.
- The underlying FlashBlade//S platform supports multi-tenancy including quality of service (QoS) guarantees. Extending that to FlashBlade//EXA to support providers who serve multiple constituencies would be straightforward.
- DirectFlash technology is a potential foundation for higher-performing, more reliable, more cost-effective data nodes than are possible with off-the-shelf components.

Pure Storage's solid technology base will enable **FlashBlade//EXA** to continue leading the field as parallel storage system technology and usage evolve over time.

Pure Storage's demonstrated commitment to sustained innovation in data storage makes it an ideal partner for AI and HPC deployments today and in the future.

1 | https://en.wikipedia.org/wiki/Clustered_file_system

2 | Now called Spectrum Scale.

3 | https://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf

4 | Pure Storage documentation and collateral use the term *data node* rather than the more generic *data server*. Both terms refer to parallel storage system components that hold data item contents.

5 | Pure Storage uses a *sizer* tool to determine the number of chassis, blades, and *DirectFlash™ Modules* (DFMs) required for a metadata cluster based on capacity and performance requirements specified by the customer.

6 | The system's underlying Purity software also supports the S3 object protocol, which is a planned future extension.

7 | https://datatracker.ietf.org/doc/html/rfc8435

8 | There is some interaction between authorities, for example when clients move data items between file system directories managed by different authorities.