Efficient interface to the GridKa tape storage system

*Haykuhi Musheghyan[1], Andreas Petzold[1], Doris Ressmann[1], Preslav Konstantinov[1], Dorin-Daniel Lobontu[1], Artur Gottmann[1], Samuel Ambroj Pérez[1], and Xavier Mol[1]

[1]Karlsruhe Institute of Technology

Abstract.
Providing high performance and reliable tape storage system is GridKa's top priority. The GridKa tape storage system was recently migrated from IBM SP to High Performance Storage System (HPSS) for LHC and non-LHC HEP experiments. These are two different tape backends and each has its own design and specifics that need to be studied thoroughly. Taking into account the features and characteristics of HPSS, a new approach has been developed for flushing and staging files to and from tape storage system. This new approach allows better optimized and efficient flush and stage operations and leads to a substantial improvement in the overall performance of the GridKa tape storage system. The efficient interface that was developed to use IBM SP is now adapted to the HPSS use-case to connect the access point from experiments to the tape storage system.
This contribution provides details on these changes and the results of the Tape Challenge 2022 within the new HPSS tape storage configuration.

1 Introduction

Grid Computing Center Karlsruhe (GridKa) is the German WLCG tier-1 center in Karlsruhe that supports four main LHC experiments (Alice, Atlas, CMS, LHCb). It is also the German regional grid computing center for non-LHC HEP experiments (Belle2, BaBar, Auger, Compass).

The demand and use of storage systems (disk and tape) is increasing from year to year, creating certain challenges in their support and maintenance for WLCG T1-T2 centers [1]. To cope with these challenges, more flexible and easily adjustable solutions are needed. Tape storage systems remain the cheapest, most reliable and secure solution for storing large volumes of data for a long term. The goal of the various T1-T2 centers is the same - to provide their users prompt access, scalable and reliable storage systems.

2 GridKa storage overview

For 20 years, GridKa has been using dCache [2] as its disk storage system and IBM Spectrum Protect (IBM SP)[3], formerly known as Tivoli Storage Manager (TSM) as its tape storage system. To meet the growing demands of the WLCG, the GridKa tape storage system was recently migrated from IBM SP to High Performance Storage System (HPSS)[4] for LHC and non-LHC HEP experiments.

IBM SP and HPSS are two different tape storage systems. Both have their specifics that require a careful study to obtain detailed knowledge. The current HPSS setup at GridKa holds to the usage of disk cache in front of tape cartridges, while this is not the case for IBM SP. Another difference is the usage of file families (FFs). A FF is an attribute of tape storage systems that is used to group or aggregate a set of files before they are flushed to tape cartridges. Using FFs allow to conveniently and efficiently organize data on tape cartridges. In both tape storage systems FFs need to be predefined. In the case of IBM SP, files are flushed to tape cartridges directly using the predefined FFs. In the case of HPSS, files are first flushed to the HPSS disk and FFs are assigned on file basis. After, they are wrapped to aggregates by HPSS itself taking into account the defined FFs, and flushed to tape cartridges. In our HPSS setup, only files in the same directory are wrapped to aggregates. Each aggregate can have maximum 100 files in it and maximum 300 GiB of size. Files that are smaller than 10 GiB are included into aggregates. Bigger than 10 GiB files are flushed to tape cartridges individually. Flushing data from HPSS disk to HPSS tape cartridges occurs every 12 hours and is easily configurable via HPSS settings.

For each experiment, a different number of FF is defined. The total number of FFs defined per experiment is based on the dataset [1] structure and naming policies of a particular experiment. The calculation algorithm is different from experiment to experiment also, due to having different dataset structure and different number of FFs. In our setup, it is calculated for every file before flushing the file to the HPSS disk. FFs are reused within the same experiment. In our setup, a single dataset can have only one FF. A single FF is assigned to a single tape drive. Therefore, a single dataset is flushed to tape cartridges using just one tape drive regardless of the size of the dataset. This setup prevents datasets from being distributed across multiple tape cartridges. Nevertheless, an exceptional case applies only for large ATLAS datasets. A single dataset that is larger than 40 TB is assigned to 8 FFs and so 8 tape drives will be used in parallel to flush them to tape cartridges. If the dataset is large, it will definitely end up on more than one tape cartridge, so we can flush it in parallel using multiple FFs, i.e resulting in multiple tape drives. Using multiple FFs for large datasets helps to avoid filling up the HPSS disk with too many flush requests, so flushing them to tape cartridges can be faster than using just one FF.

In GridKa, dCache pools are directories in a large IBM Spectrum Scale (IBM SS) [5] file system, and pool sizes are logical settings. The dCache stage pools are used to stage files from tape cartridges, and the write pools are used to flush files to tape cartridges.

In the current setup, the interaction between the dCache instance and HPSS is done using a script (see Section 4) and a software called Endit-HPSS (see Section 5).

This hybrid workflow works reliably and fulfills the requirements of various experiments. Nevertheless, there is room for an improvement in this workflow, which could lead to an more efficient, simple and more streamlined use of the setup deployed at GridKa.

This contribution focuses on an efficient interface between dCache and the HPSS tape system recently developed by the GridKa team.

3 Data migration from IBM SP to HPSS

IBM SP and HPSS have completely different tape technologies, different hardware layout and software design. To be able to switch from one technology to another, data migration needs to be done. An approach was developed to reorganize the data and check the consistency of all the data on the tape, therefore the data was copied from IBM SP to HPSS. In IBM SP, files were flushed to tape cartridges individually (without grouping, aggregating them beforehand)

[1] A dataset is a set of files located in the same directory that belong to the same data type.

using dCache *pnfsid* [2] as a filename, while in HPSS we use *HPSS logical file name (LFN)*[3], which is based on *dCache LFN* [4].

Since mid-2021, GridKa has been migrating data from IBM SP to HPSS. For this reason, the GridKa team developed a script that allows to automate the migration procedure. The migration procedure shown in Figure 1 is implemented in Python as follows:

1. An entire dataset is staged from IBM SP to IBM SS file system.
2. Information on all datasets in IBM SP like file names (logical file name (LFN)), checksum type, and checksum value is obtained from dCache database and is used for further processing.
3. Based on the dCache LFN, it generates a new LFN (HPSS LFN), by which files will subsequently be flushed to HPSS tape cartridges.
4. Calculates the FF number using a specific algorithm.
5. A dataset is flushed from IBM SS file system to the HPSS disk by specifying *necessary attributes* (see details below) and using Parallel File Transfer Protocol (PFTP) [6] for the data transfer.
6. For each file flushed to HPSS disk, a checksum verification is performed. If a checksum mismatch error occurs, then the file is removed from the HPSS disk by the script and the flushing is repeated until a successful checksum check is performed.

Figure 1 displays the data migration procedure at GridKa.

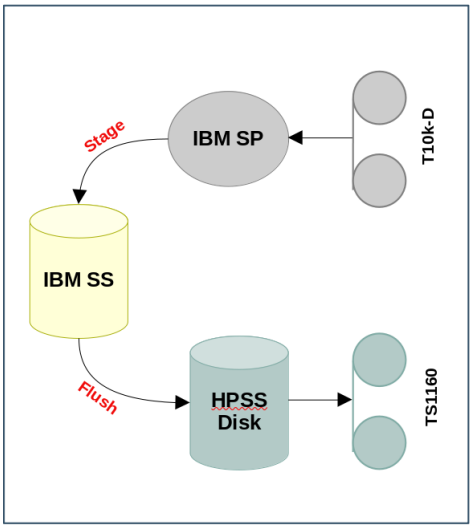

Figure 1: GridKa data migration procedure

Data migration procedure is independent from the ongoing production [5] activities and is done in the background.

[2] IBM SP LFN format: `/<vo_instance_name>/<FF>/<pnfsid>`
[3] HPSS LFN format: `/GridKa-<vo_name>/<file_path>`
[4] dCache LFN format: `/pnfs/gridka.de/<vo_name>/<file_path>`
[5] A setup that is currently running and available to various experiments on a 24/7 basis

Necessary attributes:

Several attributes need to be set when flushing a file to the HPSS disk. A few of these attributes, such as *pnfsid, checksum type, checksum value* are already set in dCache and simply passed to HPSS.

The *pnfsid* is a unique identifier in dCache and is 36 Bytes long. Specifying *pnfsid* as a file attribute allows to uniquely identify the file flushed to the HPSS disk, while only knowing LFN of the file is not enough. The file can be removed and later created again and will get a new *pnfsid* in dCache, while it has the same LFN. To avoid removing files from tape cartridges that in fact should not be removed, additional information (*pnfsid*) must be stored in HPSS as a file attribute. The presence of HPSS LFN and pnfsid guarantees the uniqueness of the file in dCache and HPSS and avoids removal of overwritten files from HPSS. If an attempt is made to overwrite a file, the pnfsid attribute is updated in HPSS.

The *checksum type, checksum value* are needed to verify the file after it has been flushed to the HPSS disk. HPSS provides a tool called *hpsssum* that can be used to calculate and validate the checksum of any file existing in HPSS disk or tape. An error code is returned when the specified *checksum type, checksum value* does not match the one computed by *hpsssum*. Checksum verification helps to avoid flushing corrupted files to tape cartridges.

Other attributes, such as *cosid, FF nummber* are pure HPSS attributes. In our setup, each experiment has its own unique identifier called *cosid*. This identifier allows to distinguish data belonging to one experiment from another. The *cosid* must be predefined in the HPSS settings. The *FF number* is calculated on a file basis (see Section 2).

All these attributes are necessary for the file to be flushed to the correct location and for the correct experiment.

4 Production setup: writing files to HPSS

Dcache allows to use external scripts to interact with the tape storage systems. The local administrator is responsible for maintaining this procedure and making it available and known to dCache.

HPSS supports user interfaces such as PFTP and Virtual File System (VFS) [7] for accessing the data it contains. PFTP is an industry-standard File Transfer Protocol (FTP) [8] extending its functionalities with parallelization. This optimizes FTP performance for flushing and staging files from and to HPSS. Another way to access and manage data in HPSS is to use VFS. VFS is a file system that is mounted to access to a local system. This allows HPSS to be available to local users and applications, just like an NFS-mounted file system. You can use standard Linux commands such as *ls, cat, touch*, etc.

Due to performance limitations in the Java Native Interface (JNI) [9] that Endit-HPSS (see Section 5) uses to interact with the HPSS API, writing files to the HPSS disk is done using a script. The script is an adaptation or extension of the script used during data migration, see 3. To flush a file from dCache to HPSS tape cartridges, the complete workflow chain of the script is defined as follows:

1. The LFN of the dCache file is obtained from the call command and converted to the LFN of the HPSS.

2. The FF number is calculated on the fly for a file based on its HPSS LFN and taking into account dataset definitions for the corresponding experiment.

3. Using a PFTP client, the file is flushed to the HPSS disk by specifying necessary attributes.

4. Afterwards, the checksum of the file copy on HPSS disk is verified with *hpsssum* tool.

5. On successful flush, a Uniform Resource Identifier (URI) [10] is returned to dCache indicating the *HPSS LFN* and the *pnfsid*.

Figure 2 displays the current production workflow between dCache and HPSS for each experiment for flushing files to tape cartridges.

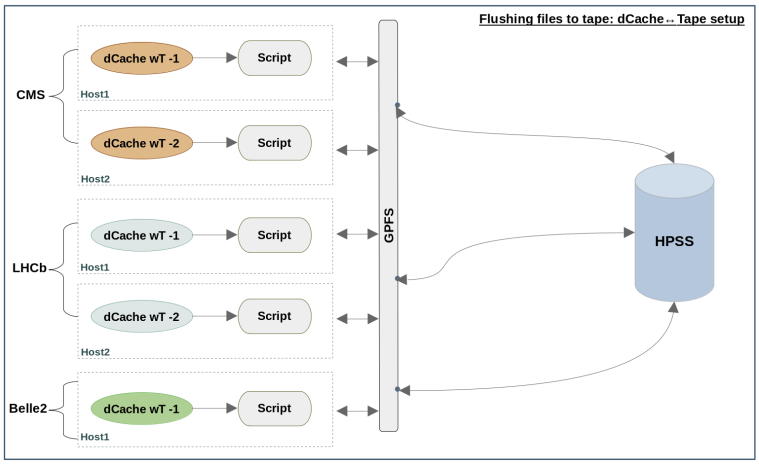

Figure 2: Current production workflow between dCache and HPSS for each experiment for flushing files to tape

The script is installed and run on the same host that is running the dCache write tape pools.

5 Production setup: staging files from HPSS

The dCache storage system provides a number of various plugins for various purposes, and one of them is a nearline storage plugin [11]. The dCache pools can copy files to nearline storage to write (“flush”) them to tape and read those (“stage”) files whenever needed. One of the nearline storage plugin is called the dCache Endit-Provider plugin [12], which interacts with the nearline storage driver. dCache communicates with the nearline storage driver by providing an appropriate request object that describes the file to be flushed, staged, or removed, and acts as a callback through which the driver reports the progress, completion, or failure of the request. Each request object has a unique ID, and dCache can cancel the request at any time using this ID.

The dCache Endit-Provider plugin is organized via various directories such as “in”, “out”, “request” and “trash” in the pool’s data directory. Depending on the specific action “stage”, “flush” or “remove”, the appropriate directory is used. The "request" directory is used to store metadata files. A metadata file is a file that contains a JSON object. The JSON object has elements/fields such as file size, time, file path and other necessary information that is used for staging or flushing from or to tape cartridges respectively. See [13] article for more details.

In the production system of GridKa, the dCache Endit-Provider and the Endit-HPSS software are used to communicate between dCache and HPSS to stage files from tape. Both are

written in Java, while HPSS provides an API written in the C programming language. JNI is a bridge between the byte code running on Java Virtual Machine (JVM) [14] and the native code written in another programming language, such as C/C++. If a file is required to be staged from tape, Endit-HPSS calls the appropriate HPSS API functions from Java code via JNI. In this case, all communication between Endit-HPSS and HPSS is done through JNI, which calls different HPSS API functions for different purposes.

The whole chain including dCache, Endit-HPSS and HPSS works as follows. If a file needs to be staged from tape, the corresponding metadata file is written to the "request" directory by the dCache Endit-Provider plugin, and dCache waits until the file is staged into the "in" directory. Endit-HPSS is a software with the following features:

- Monitoring of the "request" directory. As soon as the metadata files appear in the "request" directory, Endit-HPSS starts processing the metadata files.
- Possibility to monitor an arbitrary number of directories. This feature is important and serves for load balancing across several dCache stage pools. Dcache distributes the number of stage requests fairly/equally across the available stage pools.
- For each file, the HPSS LFN is extracted from the URI stored in dCache and used to query HPSS for the information or so-called *file attributes* stored in the HPSS. File attributes are the attributes by which the file was stored on the tape cartridge , such as tape name, position on the tape, file size, checksum and so on. Endit-HPSS queries file attributes to get the location of the file on a particular tape. The result of the query is the name and position on the tape. Files that belong to a particular tape and are part of the same aggregate have the same position number.
- Grouping of files. Files, which belong to the same tape cartridge and have the same position on it, are grouped and then compiled into lists, called aggregated list. Once the list of aggregates are ready, Endit-HPSS starts iterating through them and sequentially staging files from tape cartridges. If a failure occurs and for some reasons the file was not staged from tape, Endit-HPSS will retry a certain number of times. If the failure still persist, for example, if the tape drive is defective, then the stage request will remain active until it is fixed so that the file can be processed.
- Removal of finalized aggregated file lists after the specified/configured number of days.
- Control on the number of tape drives used for each experiment. This control is rather indirect and is guaranteed by the number of aggregated lists belonging to the same tape cartridge being simultaneously submitted to HPSS. For example, if there are 100 aggregated lists belonging to 5 different tape cartridges, then 5 tape drives will be used to stage files from these tape cartridges.
- Copying of staged files from the HPSS disk cache into the IBM Spectrum Scale (IBM SS) file system, directly into the "in" directory.

After the file has been staged with the correct name and size, the dCache Endit-Provider plugin moves the file from the "in" directory to the data pool and removes the metadata file from the "request" directory.

The dCache Endit-Provider plugin is installed for each dCache pool. As a consequence, there may be several "request" directories that will be monitored by the Endit-HPSS software, which is installed once on a dedicated host for each experiment.

Figure 3 displays the current production workflow between dCache and HPSS for each experiment for staging files from tape.

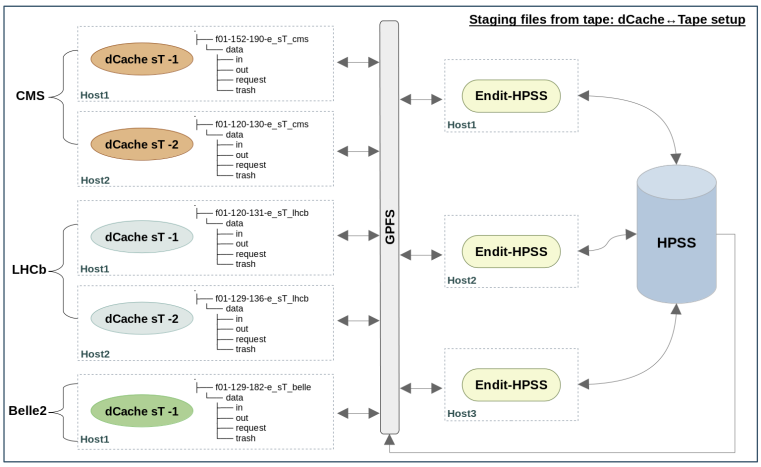

Figure 3: Current production workflow between dCache and HPSS for each experiments for staging files from tape.

6 Results and conclusions

In 2022, WLCG conducted a combined test called Tape Challenge 2022, which involved 3 LHC experiments: ATLAS, CMS, LHCb. All 3 participating experiments were expected to begin and complete testing within specific timeline. These tests consist of two main parts: the so-called "data taking" (DT) test and the "after data taking" test (A-DT). DT and A-DT tests refer to tests in which the experiment must flush and stage any data (test or real data) to and from T0-T1 centers, respectively. The timeline of these stress tests was the following:

- CMS: 2nd week of March (7th to 11th) for A-DT test
- ATLAS/CMS/LHCb: 3rd week of March (14th to 18th) for DT test
- ATLAS/LHCb: 4th week of March (21st to 25th) for A-DT test

The main purpose of these stress tests was to validate the tape throughput required to flush and stage data to and from tape storage system of T0-T1 centers during Run3. See [15] article for more details.

Since GridKa supports multiple WLCG experiments, it participated in the Tape Challenge 2022 with 3 experiments: ATLAS, CMS, LHCb. Shortly before Tape Challenge 2022, CMS and LHCb were switched from IBM SP to HPSS for tape storage. It was a good opportunity to test out the new HPSS setup, fix and adjust it as required.

Figure 4 displays the tape storage performance after switching from IBM SP to HPSS for both actions, flush and stage. During the test, an average flush throughput of ≈1500MB/s for 5 tape drives or ≈300MB/s per tape drive, within the new HPSS setup, was achieved (see Figure 4 left). For stages, an average of ≈4200MB/s for 14 tape drives or ≈300MB/s per tape drive was achieved (see Figure 4 right), which is not the case for the IBM SP use case (max ≈180MB/s per drive).

This is a very good achievement. The new setup works very well and from the experiment point of view effectively fulfills their requirements for flushing and staging files to and from tape cartridges. The new setup allowed the overall tape storage throughput has been improved by more than a factor of 2, per tape drive.

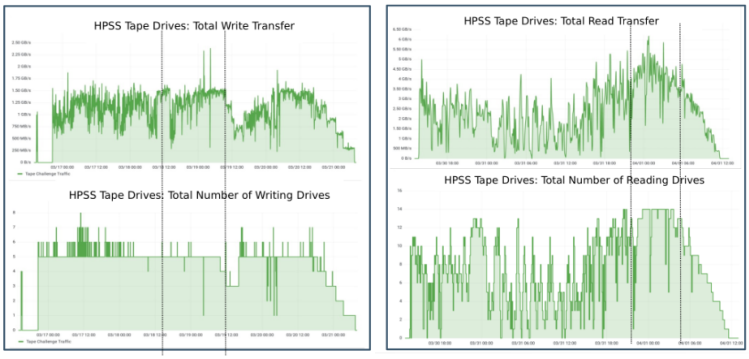

Figure 4: Tape storage performance after switching to HPSS.

Further improvements are planned on the basis of new requirements, challenges and changes that may arise over time and during the operation of a new setup.

I would like to thank the dCache developers, HPSS collaboration, WLCG colleagues, and our GridKa team experts for their contributions.

References

[1] Shiers, Jamie, *The worldwide LHC computing grid (worldwide LCG)* (Computer physics communications 177.1-2, 2007)

[2] dCache storage system, https://www.dcache.org/ (visited on 01/09/2023)

[3] IBM Spectrum Protect (IBM SP), https://www.ibm.com/support/knowledgecenter/de/SSEQVQ/landing/welcome_sseqvq.html (visited on 01/09/2023)

[4] High Performance Storage System (HPSS), http://hpss-collaboration.org (visited on 01/09/2023)

[5] IBM Spectrum Scale (IBM SS), https://www.ibm.com/products/spectrum-scale (visited on 01/09/2023)

[6] Bhardwaj, Dheeraj, and Rishi Kumar, *A parallel file transfer protocol for clusters and grid systems* (First International Conference on e-science and grid computing (e-Science'05), IEEE, 2005)

[7] Boomer, David I., *Relational database active tablespace archives using hsm technology* (Available on: http://www.hpss-collaboration.org/hpss/about/BoomerRDBMSHSM.pdf, 2006)

[8] Rahim, Robbi, et al., *Prototype file transfer protocol application for LAN and Wi-Fi communication* (Int. J. Eng. Technol 7.2.13: 345-347, 2018)

[9] Kurzyniec, Dawid, and Vaidy Sunderam, *Efficient cooperation between Java and native codes–JNI performance benchmark* (The 2001 international conference on parallel and distributed processing techniques and applications, 2001)

[10] Uniform Resource Identifier (URI), https://en.wikipedia.org/wiki/Uniform_Resource_Identifier (visited on 01/09/2023)

[11] dCache nearline storage plugins, https://dcache.org/old/manuals/Book-6.1/cookbook-writing-hsm-plugins.shtml (visited on 01/09/2023)

[12] dCache Endit-Provider plugin, https://github.com/neicnordic/dcache-endit-provider (visited on 01/09/2023)

[13] Musheghyan, Haykuhi, et al., *The GridKa tape storage: latest improvements and current production setup* (EPJ Web of Conferences. **Vol. 251** EDP Sciences, 2021)

[14] Stärk, Robert F., Joachim Schmid, and Egon Börger, *Java and the Java virtual machine: definition, verification, validation* (Springer Science and Business Media, 2012)

[15] Barisits, Martin, et al., *ATLAS Data Carouse* (EPJ Web of Conferences, **Vol. 245**, EDP Sciences, 2020)