Mass storage interface LTSM for FAIR Phase 0 data acquisition

Jörn Adamczewski-Musch[1,*] and *Thomas* Stibor[1,**]

[1]GSI Helmholtzzentrum für Schwerionenforschung GmbH,
Planckstr. 1, D-64291 Darmstadt, Germany

Abstract. Since 2018 several FAIR Phase 0 beamtimes have been operated at GSI, Darmstadt. Here the new challenging technologies for the upcoming FAIR facility shall be tested while various physics experiments are performed with the existing GSI accelerators. One of these challenges concerns the performance, reliability, and scalability of the experiment data storage. Raw data as collected by event building software of large scale detector data acquisition has to be safely written to a mass storage system like a magnetic tape library. Besides this long term archive, it is often required to process this data as soon as possible on a high performance compute farm.
The C library LTSM ("Lightweight Tivoli Storage Management") has been developed at the GSI IT department based on the IBM TSM software. It provides a file API that allows for writing raw listmode data files via TCP/IP sockets directly to an IBM TSM storage server. Moreover, the LTSM library offers Lustre HSM ("Hierarchical Storage Management") capabilities for seamlessly archiving and retrieving data stored on Lustre file system and TSM server.
In spring 2019 LTSM has been employed at the FAIR Phase 0 beamtimes at GSI. For the HADES experiment LTSM was implemented into the DABC ("Data Acquisition Backbone Core") event building software. During the 4 weeks of Ag+Ag@1.58 AGeV beam, the HADES event builders have transferred about 400 TB of data via 8 parallel 10 GbE sockets, both to the TSM archive and to the "GSI green cube" HPC farm.
For other FAIR Phase 0 experiments using the vintage MBS ("Multi Branch System") event builders, an LTSM gateway application has been developed to connect the legacy RFIO ("Remote File I/O") protocol of these DAQ systems with the new storage interface.

1 Introduction

The storage of raw data produced by a high data rate detector is a key component of an experimental facility. For the upcoming accelerator *FAIR (Facility for Antiproton and Ion Research)* in Darmstadt, a new experimental storage system is currently being developed. As a backend, this will use the Tivoli Storage Management (TSM) for magnetic tape libraries by IBM [1]. Additionally, the central *HPC (High Performance Computing)* clustre inside the "GSI/FAIR green cube" employs the *Lustre* file system [2]. So the novel storage system

*e-mail: j.adamczewski@gsi.de
**e-mail: t.stibor@gsi.de

LTSM ("Lightweight Tivoli Storage Management") [3] is intended to connect the network data stream from the experiment detectors both with TSM and with Lustre.

In 2018 the *FAIR Phase 0* beamtime campaign has been started at *GSI (Gesellschaft für SchwerIonenforschung, Darmstadt)*. Here some new technologies for FAIR are applied and tested with physics experiments at the existing GSI accelerators and detectors. In March 2019 LTSM has been used during four weeks of beamtime at *FAIR Phase 0* for production data storage of the *HADES* experiment [4]. This application delivered valuable experiences for the further development and deployment of LTSM.

2 DAQ storage use cases

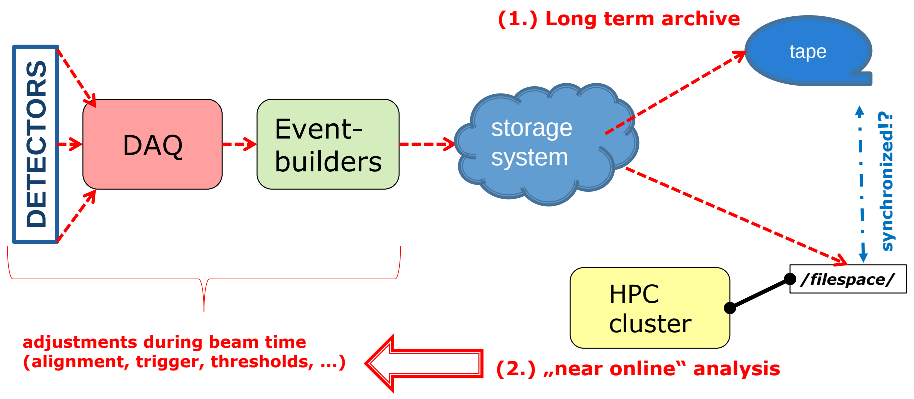

Figure 1. Use cases of a DAQ storage system: Raw data files produced by the event builders need to be written to a long term archive like a tape (1). Additionally, the same files are required at the filespace of an HPC cluster for a near online analysis during data taking (2).

When developing a storage system like LTSM one has to keep in mind the primary use cases of raw data storage from the experiment's point of view. Figure 1 illustrates this.

The detectors of a physics experiment will have any kind of data acquisition system (DAQ) producing a stream of digitized data. Usually a set of event builder processes (Event-builders) will sort these data into defined packets, according to a readout trigger number, or using a time stamp in case of a triggerless or mixed readout system. These defined packets ("Events") can be collected into files which are sent to any kind of storage system. As a trivial implementation, the storage system is just a local disk.

However, it is typically required to keep the raw data files safely for a long time (up to 50 years), which cannot be guaranteed by a single hard drive file system. Such long term archive [(1) in Figure 1] may be rather a magnetic tape library.

On the other hand, experience shows that access to these raw data is also needed for an immediate analysis during the beamtime. Experimental parameters like geometrical alignment, trigger settings, or detector thresholds, have to be controlled within hours after data taking to ensure that all adjustments are correct. Such evaluation often exceeds the simple local online quality monitoring of the DAQ readout, but requires a rather full "near online“ analysis of several files on an HPC compute cluster [(2) in Figure 1]. Because of this, the raw data files must be made available on the HPC cluster file space that is usually different from the long term archive. It is also mandatory that the files are synchronized between the archive and the HPC file space, i. e. the same files written to the archive are also available at the HPC file space for analysis within a short time (typically <1 hour after acquiring them at the detector).

3 The LTSM software

Since 2016 the LTSM [3] software has been developed at the GSI IT department as the future GSI/FAIR storage system. One of the main LTSM features is the implementation of the Lustre *HSM (Hierarchical Storage Management)* capability: files stored on a Lustre file system can be seamlessly archived to, and retrieved from, a TSM server with long-term storage media such as magnetic tapes. Therefore LTSM inherently maps the mount path on the Lustre installation with a file space path in the TSM archive.

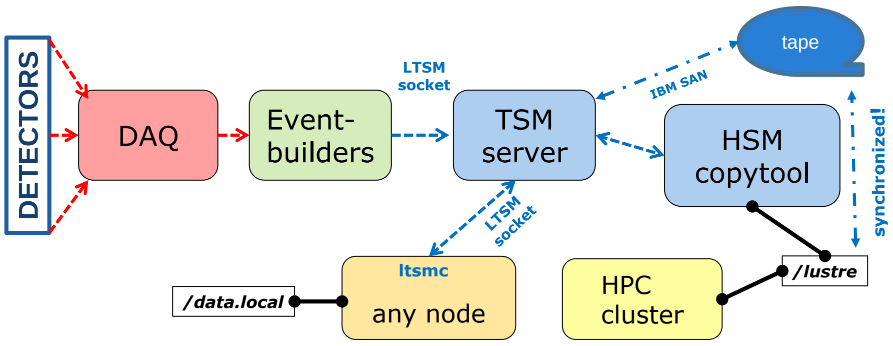

Figure 2. Components of the LTSM system: The DAQ event builders send data via LTSM socket API connections directly to the TSM server archive that is linked to a tape robot device using *IBM SAN (Storage Area Network)*. The HSM copytool provides the file synchronization between the TSM archive and the lustre file system of the HPC cluster. Additionally, an external user can access files via socket from TSM archive using the *ltsmc* commmand line tool.

LTSM mainly consists of a C library on 64bit Linux which is based on the IBM Tivoli Storage Management client library [1]. The API of the LTSM library has been used to implement connections to TSM in the scope of existing GSI data acquisition frameworks, such as DABC [5], [6] and MBS [7], [8].

3.1 C-library API and tools

The LTSM library offers an API that allows one to open a file at the TSM archive and write data into it via a socket connection. It contains the following functions:

int tsm_fconnect *(struct login_t ∗login, struct session_t ∗session)* – connect a session at the TSM server with valid login credentials

void tsm_fdisconnect *(struct session_t ∗session)* – disconnect session from the TSM server

int tsm_fopen *(const char ∗fs, const char ∗ fpath, const char ∗desc, struct session_t ∗ session)* – open a file of name fpath in file space fs with optional description desc

ssize_t tsm_fwrite *(const void ∗ptr, size_t size, size_t nmemb, struct session_t ∗session)* – write a data buffer to the currently open file

int tsm_fclose: *(struct session_t ∗session)* – close current file

This allows one to handle files in the TSM archive from user applications with a POSIX like interface.

Based on this API, the LTSM software provides a command line tool *ltsmc* for interactive or script control of the archived files from a Linux shell. It allows one to query archived files (`ltsmc -query`), retrieve files from the archive to local disk (`ltsmc -retrieve`), and write local files into the archive (`ltsmc -archive`). On top of *ltsmc* higher level scripts can be developed for more complex tasks, like comparing local and archived files, or archiving local files with different paths on Lustre.

The Lustre HSM functionalities are implemented with the LTSM library as an *HSM copytool* daemon. This executable is started on a special copytool server host that has mounted the Lustre file system, and at the same time has network access to the TSM server. By such copytoool, on request of a storage policy agent, certain files may be automatically archived to a corresponding storage node at the TSM, i. e. their data will be moved from disk to the long term storage tape. The file meta information, however, is still present on Lustre. Vice versa, the data of a file on the Lustre file space which had been previously archived out to TSM can be automatically retrieved back to Lustre on the first access of that file by a user. Moreover, it is possible to assign an already archived file to Lustre which exists in TSM first and shall be later copied to Lustre by the copytool.

Figure 2 shows the components of LTSM in the use cases of DAQ storage. The event builder software of the DAQ system can use the LTSM API to directly connect a session at the TSM server storage node, open files and write raw data via a TCP/IP socket. The TSM server functionality grants that the new files are first buffered locally and then stored to the long term tape media. The HSM copytool provides synchronization between the TSM archive and the Lustre file system of the HPC cluster. On request of a controlling daemon, the copytool will retrieve any new TSM file from the archive and transfer it to the `/lustre` mountpoint where it is available for all nodes of the compute farm. Thus both use cases discussed in Section 2 can be handled with such implementation. Additionally, by means of *ltsmc* it is possible to retrieve the archived data manually to any further node in the local network. This allows user defined data inspection and processing independent from the official analysis on the HPC cluster.

3.2 Plug-in for DABC data acquisition

The *DABC (Data Acqusition Backbone Core)* is a C++ software framework developed at GSI for general purpose data acquisition and monitoring [5] [6] . Since 2014 DABC has been used at GSI as the production event builder system of the HADES experiment [4].

To provide the new storage system for HADES, a plug-in has been developed for DABC that is based on the LTSM API described in Section 3.1. This plug-in mainly implements the `dabc::FileInterface` base class which declares POSIX-like functions, such as *fopen()*, *fclose()*, *fread()* and *fwrite()*. Therefore it matches the LTSM API very well. Since the `dabc::FileInterface` is independent of the data format, the same LTSM plug-in can be used within DABC for all existing `dabc::FileOutput` format plugins, for example HADES (.hld), MBS (.lmd), or DABC raw format (.dabc).

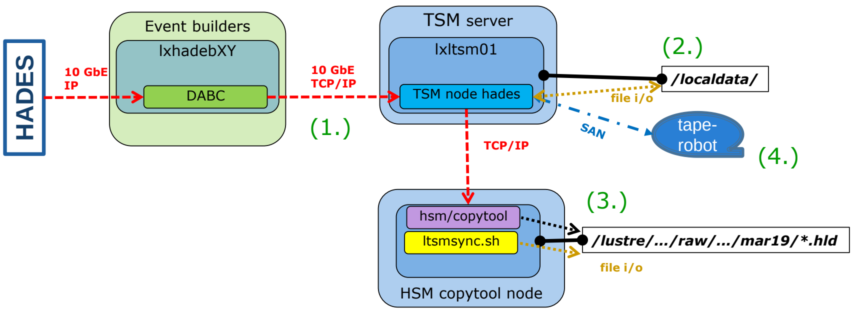

Figure 3. The functionality of the DABC LTSM plug-in as used for HADES in March 2019. The event builders can open and write hld files directly into the archive and to Lustre via a 10 GbE socket connection. See main text for details.

Figure 3 shows the usage of the DABC plug-in for the HADES data taking as provided in March 2019. The LTSM plug-in of the DABC software writes raw event data into the TSM archive (1.). The TSM server assigns the files immediately to the archive, but buffers the data first on a local disk (2.). A dedicated Lustre node runs a daemon script *ltsmsync.sh* that assigns any new file from the LTSM archive to the Lustre filespace. Thus the HSM copytool becomes active and retrieves the data of such files from archive (3.). The TSM server will physically migrate the file data to the tape robot if the local disk fill level exceeds a certain threshold (4.).

3.3 Gateway for MBS data acquisition

Several experiments at GSI used to work with the DAQ system *MBS (Multi Branch System)* [7] [8]. Such a software framework is also supported for numerous vintage platforms, like LynxOS and Power PC Linux for CAMAC and VMEbus readout hardware. Since the IBM TSM client software can not be deployed for these operating systems, the LTSM API can not be implemented directly to the MBS software.

As a workaround a gateway application has been developed as shown in Figure 4. The MBS event builder collects data from any new or legacy readout system, and sends raw data via 10 GbE sockets using the RFIO storage protocol of the previous *gStore* system [9] to a dedicated gateway server (1). Since the RFIO protocol had been already implemented in the MBS framework as a standard storage interface for all platforms, this required only minimum code adjustments. The RFIO-LTSM gateway application implements both the vintage RFIO server, and a client for LTSM. So the raw files are opened through another socket directly

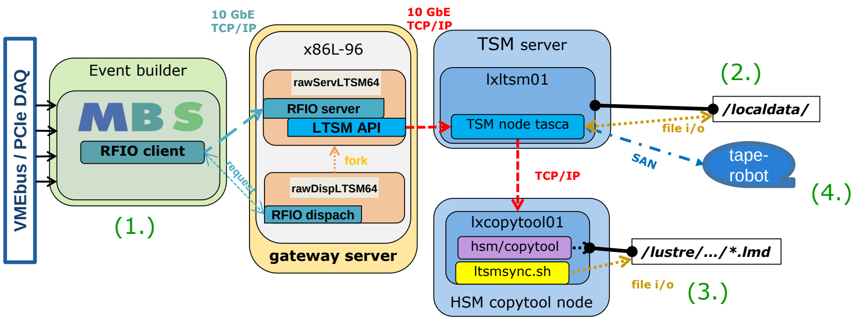

Figure 4. Functionality of the RFIO-LTSM gateway for MBS. See main text for details.

on the TSM server that receives the data at its local storage (2.). The HSM copytool and the *ltsmsync.sh* script provide the transfer of the archived files to the Lustre of the HPC farm (3.). Finally the TSM server may migrate the archived files from local disk to the long term tape robot (4.). These functionalites at the TSM server are in principle the same as for the DABC setup described in Section 3.2.

4 LTSM for HADES experiment at FAIR Phase 0

The *HADES (High Acceptance Di-Electron Spectrometer)* experiment [4] features one of the largest detector systems of GSI for almost 25 years. Morevoer, HADES will be part of the future *Compressed Baryonic Matter* research pillar of FAIR .

In March 2019 HADES performed four weeks of beamtime with an *Ag+Ag@1.58 AGeV* beam from GSI SIS18 accelerator during the FAIR Phase 0 campaign. Here the overall data rate of HADES reached about 400 MiB/s at the maximum of the 18 s beam spills. This corresponds to 16 kHz maximum trigger event rate.

For data taking, DABC with the LTSM plug-in has been used as described in Section 3.2. Figure 5 shows an overview of the HADES event building system. The subdetector data as delivered via UDP from 35 DAQ trbnet hubs [10] are collected at 5 input nodes of the DABC event building network (BNET). Then up to 12 DABC event builder nodes are writing files in parallel directly into the archive via LTSM. Simultaneously a second version of the same files is written to local RAID6 storage at the event builder servers. Synchronization between file runs is provided by an event builder master control process that automatically starts new file names on each event builder node at once after a specified amount of data (typically every 2 GB per file). Any BNET node can be monitored and controlled by a web server at this DABC master node.

A total of 380 TB of data has been recorded with this system into the TSM archive. This means an average data rate of 150 MiB/s (6.2 kHz event rate) during the entire four weeks of beam time, or about 200 MiB/s (8.7 kHz) net data rate over the actual time when data has been acquired by the HADES detectors. This rate difference mainly stems from intermissions of the accelerator beam service, and from occasional breaks of the DAQ system, both reducing

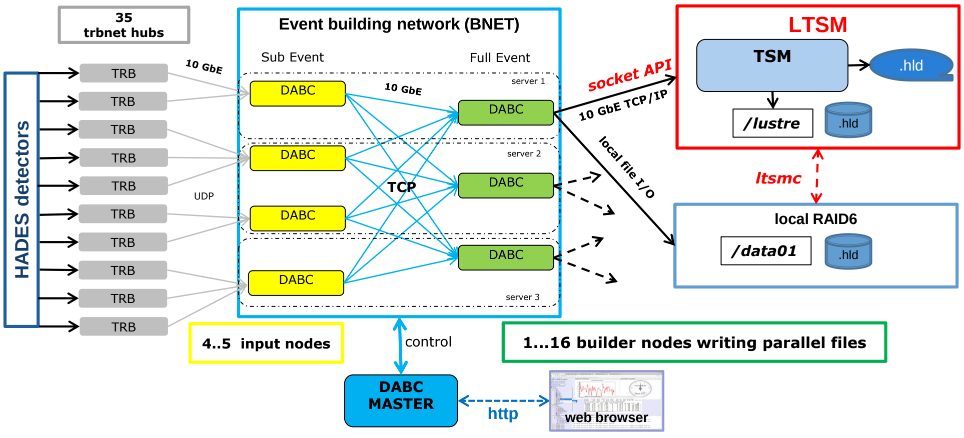

Figure 5. Components of HADES event building in March 2019. See main text for details.

the available acquisition time. The transfer time to the archive, however, does not affect this net data rate, because the event builders write files via socket directly to the TSM server. This server decouples any archiving delays from the clients, as it buffers data for the TSM node on its local disk before transferring it to tape.

In general the connection between DABC and TSM with the LTSM API worked very well. The event builder processing was not slowed down by any data back pressure of the socket connections, but could use the full bandwidth of the 10 GbE connections.

However, a rare problem has been observed when a file once was opened at the TSM node, but no data was written anymore for 2 minutes due to a stalled DAQ, or because of a missing accelerator beam. In this case, the TSM server closed down the storage connection, leading to a controlled termination of the DABC event builder process due to the error handling of the LTSM plug-in. In this case a manual operator intervention was required to restart the event building. Additionally, the file was not accounted at all for the TSM archive, so it had to be archived later by *ltsmc* from the second version on the local event builder RAID6 disk (see Figure 5). This problem was caused by a misconfiguration of a timeout parameter at the TSM server storage node and could be easily fixed.

A more severe problem showed up concerning the synchronization between the TSM archive and the HPC Lustre file system. Initially, the intended mechanism with a daemon script *ltsmsync.sh* and the HSM copytool (see Figure 3) could in fact deliver each new file from the TSM server to Lustre within 10 minutes. But as the number of files in the Lustre destination directory increased, comparing the existing files on Lustre with the TSM catalogue took more and more time for the *ltsmsync.sh*, although the checked time range was limited to the previous day only. So it then could exceed several hours until a specific file was assigned to the Lustre and the HSM copytool had retrieved its data. A crucial point was reached when the TSM server has started to migrate the data of the local files to the tape robot after the first week of beamtime. The TSM server could not handle the numerous file list queries of *ltsmsync.sh* anymore on time, as its catalogue was generally locked more and more by the tape migration process. Furthermore, in the worst case a recent file requested by the HSM copytool had been already migrated to tape, so it had to be restored back to the TSM server local disk first. These collisions of the different meta databases in Lustre and TSM led to an undesirable congestion of the intended mechanism.

In the end, the synchronization daemon and the HSM copytool had to be stopped in the second week of beamtime to prevent affecting the storage of new event builder data to the TSM server. Thus only 19 % of the about 260000 HADES beam files could be fetched from TSM to Lustre by this method. Instead of this, a workaround was set up to provide the HADES raw data files for near online analysis on Lustre: The second version of each file was copied via *nfs* and *rsync* tools from the local RAID6 disks of the four BNET event builder servers to the Lustre mount on a specific HPC cluster node. This kludge worked somewhat, although the frequent invocation of *rsync* every hour on a remote machine had a visible impact on the event building nodes: Here the kernel NFS servers exporting such file system could impose kernel waitstates, thus leading to lost UDP packets from DAQ subevents at the BNET inputs (compare Figure 5). This could cause event loss rates in the order of up to 100 Hz.

5 Outlook: the FSD server

The LTSM file API has proved to work well for directly writing event data into the TSM archive, both with DABC and with MBS systems. The command line interface *ltsmc* also turned out to be a valuable tool for user interaction between local file space and the archive node, and is continuously being improved.

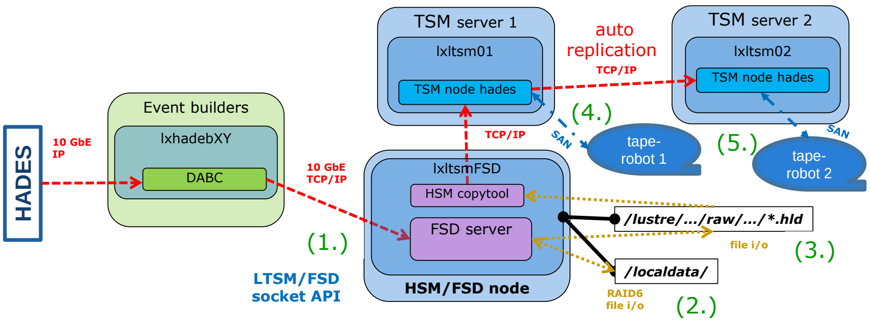

Figure 6. Event building storage with the future file system daemon (FSD). See main text for details.

However, the provided mechanisms to make the archived files available on Lustre for near online analysis are not sufficient for data rates and file numbers as produced by experiments like HADES. Although *ltsmsync.sh* daemon and HSM copytool do work for smaller atomic or nuclear physics experiments using MBS, they will not be a suitable solution for future FAIR detector systems collecting multiple amounts of HADES data.

Because of this, another component is going to be implemented for the LTSM system. Figure 6 shows how the newly developed *FSD (File System Daemon)* software is intended to work for event builder storage: The event builder process does not use the plain LTSM API socket for connection with the TSM server, but opens the file via another socket protocol at a specific FSD server node (1.). The FSD server process will buffer the event data as a local file first (2.), and will then copy it to the directly mounted Lustre file system (3.). Subsequently the HSM copytool will be invoked for each file to transfer it into the TSM archive of the associated TSM Server (4.). Finally the TSM node will migrate the data to the tape robot, and may replicate it automatically to a second TSM node (5.).

So in this scheme the data flow is always in the same direction. Moreover, it is not necessary to check frequently the meta data of the TSM or Lustre for files to be retrieved, since the FSD server manages all data transfer jobs with an internal work queue of states for each file. What is more, the FSD client API does not require the IBM TSM libraries anymore, since the actual connection to the TSM server is done by the FSD server node. Therefore it may be implemented into event builder software also for platforms that do not support the TSM client.

A first version of such FSD server software has been developed and is available at the LTSM repository [3]. Within four days in January 2020, about 140 TB of randomly generated event data files have been successfully transferred from the HADES event builder servers via the FSD server to Lustre, and further into the tape robot.

It is expected that the FSD solution is deployed for production at GSI until summer 2020 and will be applied at the next beam times.

Acknowledgements

The LTSM project has been funded by Intel through GSI's Intel Parallel Computing Centers.

References

[1] C. Brooks, P. McFarlane, N. Pott, M. Trcka, and E. Tomaz: *IBM Tivoli Storage Management Concepts* (IBM Press Books, Redbooks, 2006)

[2] E. Barton and A. Dilger: *Lustre*, in *High Performance Parallel I/O* (Chapman & Hall/CRC Computational Science, 2014) 91-106 pp.

[3] T. Stibor: *The LTSM project* (online), https:/github.com/tstibor/ltsm

[4] The HADES collaboration homepage (online), https://www-hades.gsi.de

[5] The DABC homepage (online), http://dabc.gsi.de

[6] J. Adamczewski-Musch, N. Kurz, and S. Linev: *Developments and applications of DAQ framework DABC v2* , J.Phys.Conf.Ser. **664** (2015) 082027, doi:10.1088/1742-6596/664/8/082027, https://iopscience.iop.org/article/10.1088/1742-6596/664/8/082027

[7] The Multi Branch System homepage (online), http://daq.gsi.de

[8] H. G. Essel, J. Hoffmann, N. Kurz, and W. Ott: *The general purpose data acquisition system MBS*, IEEE TNS **47**, No.2, (2000), pp 337-339

[9] H. Goeringer, M. Feyerabend, and S. Sedykh: *Experiences with gStore, a scalable Mass Storage System with Tape Backend*, J.Phys.Conf.Ser. **119** (2008) 052018, doi:10.1088/1742-6596/119/5/052018, https://iopscience.iop.org/article/10.1088/1742-6596/119/5/052018

[10] J. Michel et al: *The upgraded HADES trigger and data acquisition system*, JINST **6** (2011) C12056, doi:10.1088/1748-0221/6/12/C12056, https://doi.org/10.1088/1748-0221/6/12/C12056