PROCEEDINGS OF SCIENCE

ERADAT and DataCarousel systems at BNL: A tool and UI for efficient access to data on tape with fair-share policies capabilities

Jérôme LAURET[1]
Brookhaven National Laboratory
Upton, NY 11973 – USA
E-mail: jlauret@bnl.gov

David Yu
Brookhaven National Laboratory
Upton, NY 11973 - USA
E-mail: david.yu@bnl.gov

PoS(ACAT2010)023

[1] Speaker

PROCEEDINGS OF SCIENCE

The BNL computing facility, supporting the RHIC experiments as its Tier0 centre and later, the US based Atlas/LHC Tier1 centre, had to address at a very early stage the issue of efficient access to data stored to Mass Storage System. On such tape based archival storage, random access destroys performance by causing too frequent, high latency and time consuming tape mounts and dismounts. Coupled with a high job throughput streaming from multiple RHIC experiments as data is acquired and other datasets processed simultaneously, in the early 2000, the experimental and facility teams were lead to consider ingenuous approaches to restore efficiency the destroyed by the increasing demand and request pattern complexity. A system for tape access inspired by a "batch" system approach was developed and integrated to the job production system. Based on the initial OakRidge National Laboratory (ORNL) Batch code, its purpose was to organize restore from tape for data production accounts. In parallel, a highly customizable layer and UI known as the DataCarousel was developed in-house to provide multi-user fairshare with group and user level policies controlling the sharing of resources amongst users. The simple UI, based on a perl module, allowed to create user helper script to restore datasets on disks as well as had all the features necessary to interface with higher level storage aggregation solutions. Hence, beyond the simple access at data production level, the system was also successfully used in support of numerous data access tools such as interfacing with the Scalla/Xrootd MSS plug-in back end (STAR) and similarly, the dCache back end access to MSS. Today, all RHIC and Atlas experiments use a combination of the Batch system and the DataCarousel following a 10 years search for efficient use of resources. In 2005, BNL's HPSS team decided to enhance the new features such as improve the HPSS resource management, enhance the visibility of real-time staging activities and gather statistical of historical data for performance analysis. BNL Batch, aka ERADAT, provides dynamic HPSS resource management and scheduled read job efficiently while the staging performance can still be further optimized in user level using the DataCarousel to maximize the tape staging performance (sorting by tape while preserving fair-shareness policies). In this paper, we will give an overview of our system and its development cycle and share the findings of our efforts.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research - ACAT 2010
Jaipur, India
February 22–27 2010

PoS(ACAT2010)023

1. Introduction

When storing large amounts of data, tape can be substantially cost effective (in terms of the media cost, power consumption, and air conditioning cost), compared to modern storage technologies such as hard-disk or other data storage devices. Therefore, tape storage is still commonly used in large computer centers, primarily as a high capacity medium for backups and archiving.

The BNL [1] data center, hosting the RHIC [2] and Atlas [3] Computing Facility (RACF) holds near 15 PB of data on tapes, serving science researchers from both RHIC and the LHC/US-Atlas collaborations. It is operated by a system called HPSS [4], a software stack able to manage Peta-Bytes of data on disk and robotic tape libraries. The facility serves as the Tier0 centre for RHIC and as a Tier1 centre for Atlas and is equipped, amongst other hardware, of six Sun/TSK SL8500 each able to support up to 5 PB of data.

1.1 Problematic

Tape technologies and tape access are inherently sequential. As such, collaborations have put a great deal of thoughts into how their data is saved onto tapes and how to optimize data mining and data production workflows. From a single production account perspective for example, this implies taking into account the time sequence and ordering of files on tape when reading them back and ordering job sent to a queue system accordingly. However, this simplistic approach becomes problematic if one has to produce or mine datasets from different period in time, the stochastic nature of the workflow causing an access pattern forcing tape mount/dismount to satisfy all requests. The problem is exacerbated if there is a real need for users (one to two order of magnitude more access pattern complexity) to access data on tape. Since tape storage are so much cheaper and disk buffer still limited, the tape storage system in fact is being used as a near real-time random access device. This means user(s) may be staging (restoring) any number of files out of any random tape at any time, 24 x 7.

1.2 Technology issues overview

Tape access being sequential in nature, one may fast forward or backward but optimal access would be achieved if one could access all files from one tape sequentially without having to dismount it ever. Tape systems are good for archiving, but not for reading because of the long latency for random accesses: whenever there are lots of tapes queued up for staging, the bottleneck is the limited number of tape drives. Furthermore, the source of tape access latencies can be indentified as (a) the time it takes to transport the tape inside the library (b) the mount time (c) the position to find/seek a file on a tape (d) the rewind and dismount time (e) the number of tape marks (or separators between blocks on the media). An efficient system would need to consider all of those aspects to resolve the problem at hand.

1.3 Tools developed at BNL and timelines

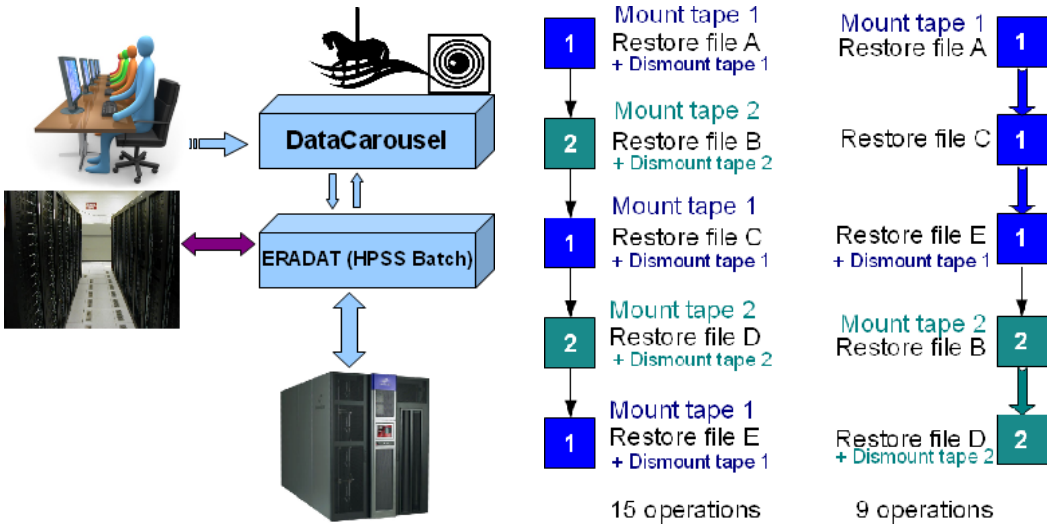

Figure 1: ERADAT (*Efficient Retrieval and Access to Data Archived on Tape*) and the DataCarousel relative interdependence. ERADAT sits at the lowest level, interfacing directly with the HPSS API and acts as a queuing system. The facility production jobs may directly interact with it. Users or high level services typically interact with the DataCarousel, implementing advanced features such as fair-share and resource handling policies. The system essentially allows minimization of tapes mounts/dismounts as seen and illustrated on the right hand-side panel.

In the early 2000, the experimental and facility teams were pushed to consider ingenuous approaches to retrieve files from mass storage during the data production workflows. A tape access "batch" system integrated to the production system was first developed, based on the initial OakRidge National Lab (ORNL) Batch code. The system could be described as a DAG, whereas the job would wait for its requested file before moving to the processing stage but not before a load condition would be reached. Since data production were usually as a sequence of jobs ordered in time (hence more or less accessing the same tapes at any given time), this level of optimization was sufficient and the order of job processing was driven by the restore of files from tapes (as decided by the HPSS API calls when a list of files was requested). In parallel, a highly customizable layer and UI known as the DataCarousel was developed in-house to provide multi-user fair-share with group and user level policies controlling the sharing of resources. The tool was delivered in 2001 and has showed great success in allowing users and working group to restore in a coordinated manner datasets from mass storage on live disks depending on their physics needs.

As the increasing demand of staging files from tapes as well as the new tape drive technologies were added to the system, the initial version of Batch could no longer handle the diversity of hardware as well as new requirements suggested from users not easily handled by the DataCarousel alone. Such features included treating the biggest request queue first (load the tape with most files requests first) which could not be specified through the HPSS native API alone. In 2005, the DataCarousel was also used as a back-end to Scalla/Xrootd by the STAR experiment [5] and the system drove even more requirements such as request lifetime expiration

and fine grain control at class of service level. By the end of 2005, the BNL RACF HPSS team worked on enhancing the "HPSS Batch" system, in order to provide better performance and resource management. In 2010, reaching full and demonstrated maturity and stability over years of production mode operation, the new Batch was renamed to ERADAT, standing for *E*fficient *R*etrieval and *A*ccess to *D*ata *A*rchived on *T*ape. The overall relation between the DataCarousel and ERADAT is illustrated in Figure 1. The right side panel illustrates a simple case of how, by ordering requests, one can reduce fifteen operations to nine operations and saves mounts and dismounts overheads. In the next sections, we will describe the systems in more details with illustrative and factual examples taken from real usage accounting.

2. ERADAT and the DataCarousel

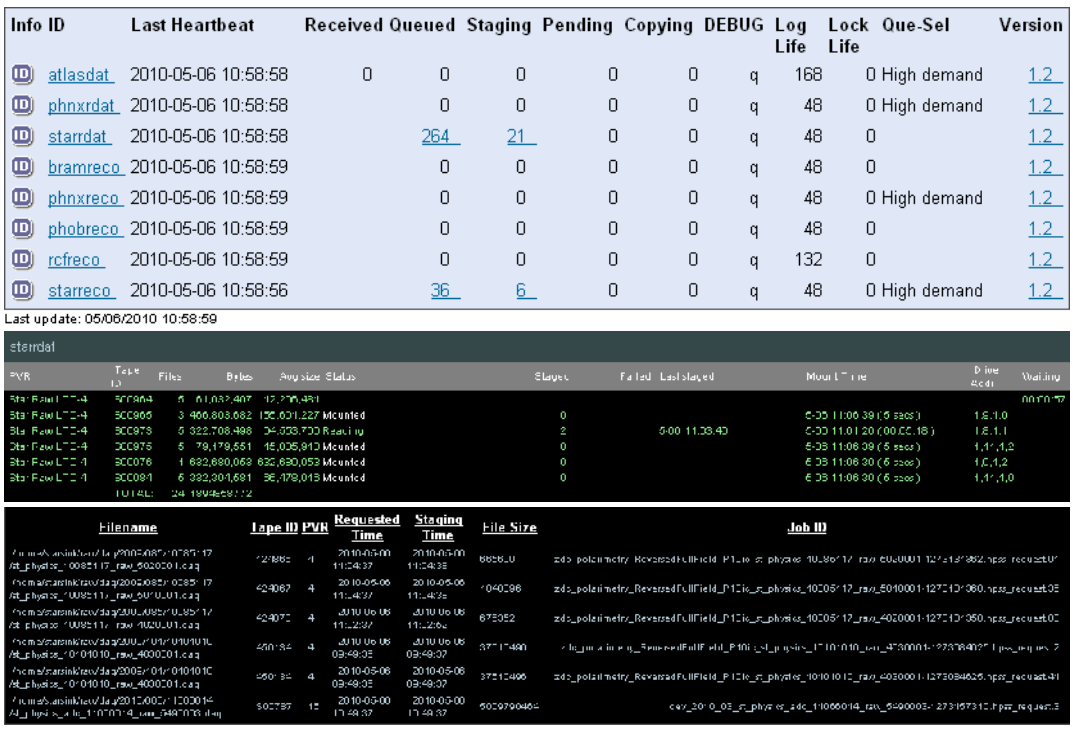

Figure 2 : ERADAT monitoring snapshots: From top to bottom, overall status window, tape mount detail status window and file list progress and status. Additional panels not represented here allow monitoring cartridge mount counts, error reports and job summary (success, failures) or generate read performance charts.

2.1 ERADAT

The *E*fficient *R*etrieval and *A*ccess to *D*ata *A*rchived on *T*ape or ERADAT, is a file retrieval scheduler for IBM High Performance Storage System (HPSS). ERADAT evolved from

the Oak Ridge batch code[2] customized to BNL's requirements. ERADAT has evolved to include many additional features such as dynamic drive usage allocation, support for multiple-projects and groups, support for multiple drive technologies (9940 and LTO drive series), request lifetime expiration and basic policies for file restore (namely FIFO, high and low demand first). ERADAT also keeps all transaction history for performance reporting purposes but the historical usage also helps guiding and fine tuning the system. ERADAT allows modifying the drive allocation on the fly via a Web-based monitoring system and control interface (as seen on Figure 2).

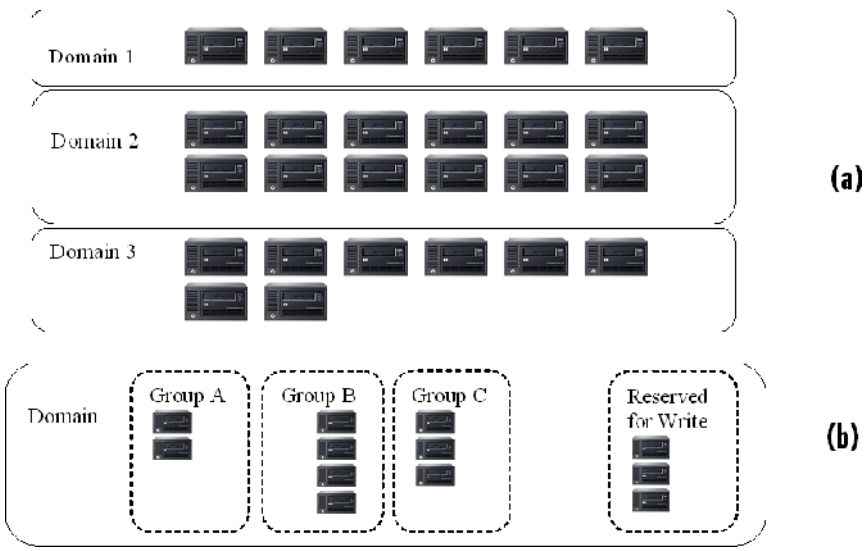

Figure 3: ERADAT allows separation of resources by domains and domains can be further partitioned and allocated between groups to the granularity of users.

ERADAT has several level of optimization and rests on a few key principles addressing the tape technology limitations listed in section 1.2 as well as practical organizational realities:

- When thousand of requests are queued for staging, there is a need to have a way to handle special higher priority requests without infinite waiting. In other words, requests marked as high priority will be processed by separate queue, and then sorted by the same Staging Algorithm (by demand, or FIFO). This feature allows such requests to take the next available drive without waiting.
- Equipments are purchased by funding from each individual experiment (STAR, Phenix and US-Atlas). Each experiment must have a dedicated number of drives available for their users and corresponding to their respective allocations. ERADAT is designed to support multi-domain partitioning of equipment in order to guarantee the resource availability at all times (see Figure 3.a).

[2] To our knowledge, this project and initial code was not documented and the work not published.

- Each experiment may have multiple groups of users and sometimes, they want to have resource reservation for the end users (fro example, group A may have more priorities than group B). ERADAT can create virtual partitions between groups to throttle the tape-drive usage by group. The partition can be dynamically re-adjusted without interrupting any running process. It is very important that we constantly have to watch the read and write traffic and fine tune the resource allocation so the drives can be fully utilized. In Figure 3.b, there are a total of 12 drives available for this Domain. The drives are logically divided into 4 partitions; 3 drives are reserved for writing. Group A can only use up to 2 drives. Group B can use up to 4 drives. The allocation is based on the agreement made between the groups. If Group A needs to borrow drives from Group B, we can always adjust the partition without any service interruption. This feature is mainly used to separate data production, massive restore (Scalla/Xrootd requests for example) and individual user requests.
- Each experiment may have multiple types of drives, such as 9940B, LTO-3 and LTO4. ERADAT has a virtual Physical Volume Repository (or PVR) that is very similar to HPSS's PVR – tapes are scheduled by PVR, so the tape-drives usage can be efficiently used. Each PVR has a dedicated manager thread for scheduling task as seen on Figure 4. Also, any scheduler can be locked at any time: for example, if the 9940B PVR needs to be down for maintenance, the LTO4 schedulers can still continue to serve files. All new requests/jobs for 9940B tapes will be queued in memory
- Tape has long latency for random accesses since the deck must wind an average of one-third the tape length to move from one arbitrary data block to another [9]. For example, HP LTO-4 media can do 120 MB/s native data transfer, but the access time may take up to 62 seconds from the beginning of the tape, and it takes 124 seconds to rewind from the end of the tape. This means that each tape mount requires at least 117 second plus the actual read time. Clearly, the tape mount is the real performance bottleneck. To cope for this problem, ERADAT sorts requests by tape cartridge and file position, so that all the requests on the same tape can be read at once sequentially in order to minimize tape mounts. ERADAT implements FIFO and "by demand" policies (more advanced algorithm are handled by the DataCarousel) and they may be enabled on the administrator request. We will discuss the relative merits of handling this at ERADAT or DataCarousel level in the next section.

Figure 4: ERADAT supports multiple drive and tape technologies simultaneously.

2.1.1 Advanced features

ERADAT has more features we will quickly list:

- Error recording: every request consumes resources, and if it leads to an error, further request may consume resources for no purpose. ERADAT hence implements a mechanism for handling some failures in real-time. ERADAT fails the request immediately under the following 3 conditions (1) the request is invalid (bad filename or file is not in HPSS) (2) tape is locked in HPSS or known to be invalid (broken or lost) (3) LSM is down.
- To limit tape mounts, ERADAT via the HPSS API also checks if the file is on the HPSS disk cache. If so, the file will be marked as "staged successful" immediately without delays.
- Callback feature: ERADAT calls a "callback script" that is provided by the experiment setup when the file is staged. The callback script should know how to deliver the file upon availability. Each file maybe handled differently and that is totally up to experiment's own preference and coding logic. This is the main mechanism used by the DataCarousel to finalize requests for staging.
- Retry policies: ERADAT implements error retries. The retries are handled by a table and policy set by group. For example, group A may decide that upon HPSS error -5 (I/O error), the system has to retry 5 times and upon error -16 (Resource Busy) it would retry up to 10 times before abandoning.
- Dead job monitoring: due to hardware problems, jobs may hang indefinitely and no longer appear as active. In such a case, an Email is sent to the administrator.

2.2 The DataCarousel

The DataCarousel is an extendable and fault tolerant policy driven framework and API allowing, in a multi-user environment, for collaboration to make requests for files archived in a Mass Storage System (HPSS) and have all requests managed and coordinated the same way a full-fledge "batch system" would. The DataCarousel is composed of two separate components, a server and a client, and a few other tools such as a Web interface for monitoring supplement the system with minor additional functionalities. The DataCarousel is entirely written in perl and a SQL based database storage in the back end. The scripting language features of perl are highly leveraged to provide plug-and-play features we will summarize.

The client is a thin script which sole purpose is to add requests to a central database. The server is the heart of the system and sorts the records, creates a job of N file to retrieve and submits that job to ERADAT as a bundle of requests according to policies. ERADAT call back feature would then call a DataCarousel handler after the file appears on HPSS cache. The handler purpose would be to pull the file out of cache on behalf of the user and update the primary request status accordingly (depending on success or failures). The implemented default policies include:

- NONE: no policy is applied; unless other options are applied, FIFO will be applied.

- EQUAL: all users are provided equal shares of the resources (in essence, each user has its own FIFO)
- GROUP: users are arranged into groups and all groups are provided equal share – groups are loosely defined and users may be sorted in usage scope such as "data management", "production" or "user" (the default). Groups have also been defined based on physics topics (in this later case, the usage is based on an honored contract as switching between groups is based on the client interface command line parameter).
- GRPW: same as the previous policy but each group is assigned a weight.

The policies are not necessarily exclusive – for example, if the policy is set to GROUP and the system detects that only one group is available, it may fallback to the EQUAL policy. Additionally, the record may be sorted depending on an ORDER option. Whenever ORDER=NONE is applied (and let us imagine our fairshare policy is NONE), the processing will be strictly FIFO while in ORDER=TAPEID mode, the DataCarousel will submit requests, reshuffling their temporal arrival based on their position on a tape. Unlike the mechanism implemented in ERADAT, the DataCarousel is more flexible and can be tuned for performance or fairshareness. In ERADAT, we would like to note that the "by demand" sorts the requests by the most requested tape first, then going to the lowest. However, in this mode and in constant request feeding scheme (always the case in full operation mode), this approach will lead to resource starvation: that request for a single file from one specific tape may never be satisfied. In the DataCarousel, there are however multiple options to avoid this scenario but if used, ERADAT should be set to not use sorting to avoid clash:

- In its default option, records are strictly ordered by tape ID. An example of this is that if record X is to be treated and belong to tape A, all requests belonging to tape A will be considered for treatment to avoid dismounting the tape and lose mount/dismount time (see Figure 1). This may however lead to un-fairness as it may create conditions where a long time may occur before a later request belonging to tape ID (l) > tape ID (k) could be satisfied especially if more requests of type (k) comes after (l).
- The SOFTID algorithm considers time then tape ID over all requests. This algorithm clearly does not fully optimize on a strict tape ID sorting but avoids the side effects as described above. It may however lead to less efficient restores.
- The TSLIDER option is a compromise option: it is implemented as a time slider based sorting where an old record for one file, even if alone on a tape, will soon be submitted as a request. In other words, in this mode the older the request, the higher the chance to be considered, regardless of how frequent the tape ID appears in the un-treated full list of pending requests and records.

It is to be noted that in the work presented in reference [5], the authors had studied multiple other fairshare mechanism including FIFO and combinations of different weights to the tape sorting, EQUAL or GROUP mode as well as considering past historical usage. In this

reference, it was found that the best approach was to consider a weighted faire share grouping leading to the best quality of service, good throughput and the lowest delays for the users. It achieved in other words, full fairshareness.

The DataCarousel easily allows extending the SHARE policies using a simplistic yet very flexible mechanism. Any new method BLA could be implemented as a stand-alone perl script named ShareBLA.pl and, leveraging evaluation capabilities of perl. The code would be executed within the core policy routine, the code would have to follow and use several preset variables accessible globally and return true/false upon success/failure (the documentation guide the developer on how to implement and what to operation are expected).

Many of the features now implemented at a lower level in ERADAT were set in the DataCarousel as the initial batch system (aka ORNL batch) did not provide such features. For example, error handling is implemented and has a versatile set of retrieval and failure recovery: retries are done for HPSS errors, client errors, storage space errors upon file retrieval to actual physical disk and miscellaneous communication and network problems. The retries upon HPSS failures are not handled identically to ERADAT: a re-submit of a failed record would only happen upon a bootstrap period (for example, once 12 jobs). The assumption is that immediate retires may lead to persistent error while a later one may help relieving transient errors. The system also detects slow HPSS responses and throttles as need be (reactive on feedback). Throttling is also automated if network communications are detected: essentially, if the queue of requests is not absorbed within the expected time average, the system will cut down on the number of files to be requested at the next job until the system stabilizes – it will then revert to the initial expected submission rate and try again the "cruising" speed. Emails notices are sent and optimal system parameters "hints" are sent to the DataCarousel manager. With all its features, it is nearly self-operating and self-recovering since 2007 and its easy interface has allowed the STAR experiment to interface Scalla/Xrootd file retrieval to this system for achieving coordination of file requests from MSS to distributed storage.

3. Results

To illustrate the benefits of our approach, we have selected three main examples (a) a usage in Data Mining scenario at RHIC (STAR data reconstruction Job Processing) (b) an ESD re-processing case at US-Atlas and (c) a mass retrieval of files for distributed storage population in STAR. In the next sections, we will make several assumptions to illustrate the gain and speed benefits. Especially, all base comparison calculations rest on hardware specifications provided by the manufacturer. It is to be noted that actual performance may vary, due to other ambient factors such as tape's media condition, tape's delivery time (location), file size and many other factors.

HP StorageWorks LTO-4 Ultrium 1840 Tape Drive - Specifications

Component Specifications

Specifications	Component	LTO-4 Ultrium 1840	LTO-3 Ultrium 960
Performance (1 Kb = 1,000 bytes)	**Native Data transfer rate**	120 MB/s with LTO 4 media	80 MB/s with LTO 3 media
	Data rate matching range	40-120 MB/s	27-80 MB/s
	Data access time (from BOT)	62 seconds typical for LTO 4 media	53 seconds typical for LTO 3 media
	Rewind time from EOT	< 124 seconds for LTO 4 media	< 98 seconds for LTO 3 media
	Rewind tape speed	7 m/s for (LTO 3 and LTO 2 media)	7 m/s for (LTO 3 and LTO 2 media)
	Average load time	< 19 seconds (RW)	< 19 seconds (RW)
	Average unload time	< 19 seconds (RW)	< 19 seconds (RW)

Figure 5: Performance specifications for the LTO-4 and LTO-3 tape drive and technology.

3.1 Data Mining at RHIC

Our first use-case is based on the data processing at RHIC. In this case, we will be using ERADAT with the optimization option and sorting "By Demand". STAR reconstruction jobs have 18 LTO-3 drives at its disposal. On 2009/02/03, the data processing requested 575 files spanning over 15 tapes. Concentrating on a single tape #409167, using ERADAT we observed that this tape was mounted only twice during the time period involved in the processing – those tape operations restored 58 files with average file size of 441 MB each. The files from that tape would have arrived (if un-ordered) in 32 sequences that is, interrupted by requests from other tapes (or 32 bundles). The overall performance using ERADAT is represented on Figure 6 and peaked at 515 files / hour.

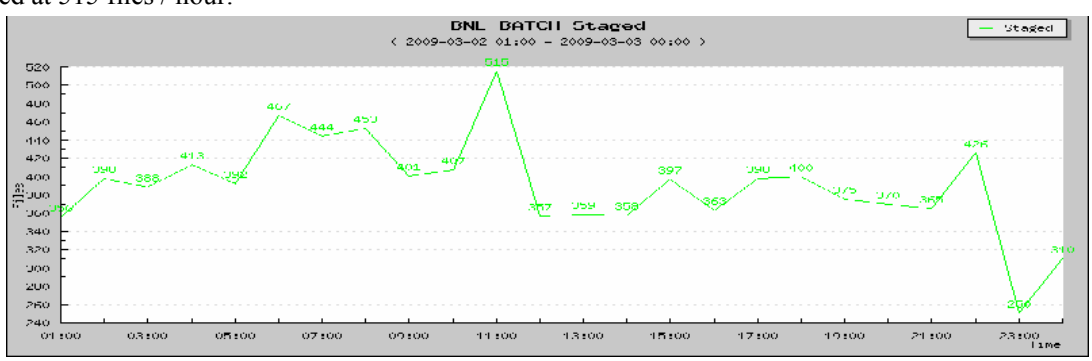

Figure 6: File retrieval performance for the RHIC/STAR data reconstruction use-case.

Without tape sorting, 32 bundles and 58 files requested from a single tape would have caused 32 mounts (if strict FIFO mode) and a restore at speed of no more than 1.8 files per mounts. Using the manufacturer's parameters from Figure 5 and considering:

- A tape delivery time of 5 sec

- Mounting (loading): 19 sec
- Positioning on file location (assumed to be in the middle of the tape in average): 53 / 2 = 26 sec
- Actual data transfer: 1.8 files * 441 MB / 80 MB/sec = 9.9 sec
- Rewinding the tape:: 98/2 = 49 sec
- Dismount (unload): 19 sec
- Place the tape back into the library: 5 sec

The total time would be 132.9 seconds per mount hence a total time of 71 minutes for 32 mounts with for net effect (for ~ 25GB of data) of a maximal effective restore speed of 6 MB/sec. With ERADAT, the same calculation and reasoning and the observation that the tape was mounted twice only, would however lead to a data transfer of 29 * 441 MB / 80 MB/sec = 160 sec time with the same overhead than previously considered. 58 files over two mounts would then be restored in 10 minutes with an average restore speed of 46.16 MB /sec hence a gain by a near factor x8. Figure 7 illustrates the overhead to actual ration before/after sorting.

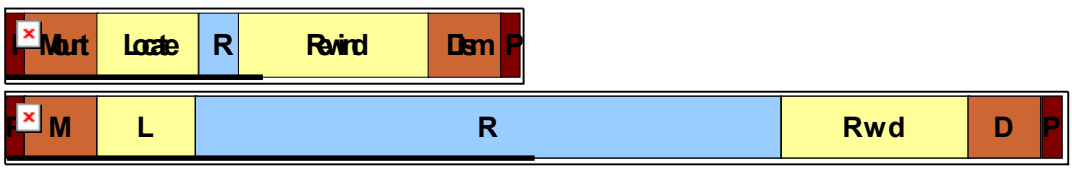

Figure 7: Overhead time is all but the blue section of the bands above for (top to bottom) FIFO and optimized (tape sorting) mode. In the case of FIFO, the overhead time exceeds by far the time to restore files on table illustrating the need to recover the maximum amount of files from a given tape. In our example, we went from 6 MB/sec to 46.2 MB/sec restore speed by bundling requests from the same tape together.

3.2 ESD processing at the LHC/US-Atlas

A similar use case was studied for the LHC/US-Atlas ESD re-processing. The case also involves the "By Demand" optimization option using a combination of 10 LTO-3 and 17 LTO-4 drives (superior performance data sheet over the LTO-3 drives) in a mixed drive technology scenario. The overall performance of our case study is represented on Figure 8.

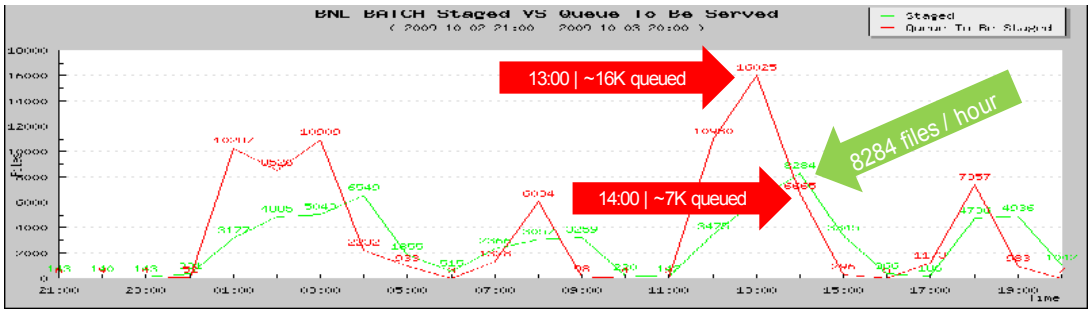

Figure 8: Performance graphs for US/Atlas ESD processing based on the 2010/03/09 case. Performance reached a near 83 k files per hour restore rate and 225 GB / hour with a 27 MB file size.

In this use case, the careful observation of a single tape #500425, we saw that 2,277 requests were on this tape and they were received in 530 different bundles. With ERADAT's high-demand optimization, tape #500425 was only mounted 3 times, staged 2,277 files, 77 GB of data with its own average file size of 34 MB. Since the files arrived in 530 different timestamp (or bundles) and if processing in FIFO mode, we would have ended up with 4.3 files per bundle and 530 mounts. Following a similar argument than in the previous section but folding in the LTO-4 performance from Figure 5, we would infer a data transfer of 1.22 sec for 4.3 files over an overhead of ~ 141 sec total. Hence, and over 530 mounts, the total time for restore would be 21 hours and the data transfer speed would be at a maximum disastrous 1 MB/sec (extremely low for a LTO-4 drive). However, with optimization from ERADAT alone and a sorting by tape, we ended with 3 mounts hence a total of 356 seconds per mount for 759 files per mount. This means we reduced the time of restore to 18 minutes and boosted the restore speed to 73 MB /sec.

In this case, the change from 1 MB/sec to 73 MB/sec represents more than an order of magnitude speed increase in file restore from tape by using ERADAT. Coincidently, the BNL US/Atlas Tier1 data center has been the site with the fastest job processing rate to date.

3.3 DataCarousel file restore for distributed storage at STAR and discussion

Our last use case is a massive file restores using ERADAT in FIFO mode and allowing the DataCarousel to use the tape ID file sorting. The statistics was based on an exercised performed on 2010/02/04 involving 15 LTO-3 drives (with an overlap shared by data production) where 7,187 files were restored over 106 tapes for a total of 4.4 TB. In average, the tapes were mounted 1.21 times instead of an average of 2 in the ERADAT example of section 3.1 leading to an even greater performance. However, this raises the question: why not the asymptotic value of once only?

For one thing, the DataCarousel uses tape ID as they are made available: each request trigger a lookup for its associated tape ID and this is done in a separate process and asynchronously to request processing. This MetaData lookup is slow and the initial few first job submission to ERADAT may then not be sorted by tape ID (as the information may not yet be fully available). This is a minor impact and only goes for the first 30 mnts or so for this amount

of records. But more importantly, the resources were shared by other processes and HPSS certainly introduces multiple foreign elements in our improvement strategy not all under our control:

- HPSS does not dismount tape immediately after the last staging because HPSS reserves a period of waiting time in case another staging request for that tape comes along. In resource sharing mode, this may cause additional delays (while this delay is experimentally observed and inferred, the manufacturer does not specify it and HPSS acts as a black box). According to our monitoring records, HPSS may wait up to about 15 minutes to dismount an un-requested tape after the last read while other requests are pending (we do not understand what is behind this "feature").
- Correlatively, we observed un-explainable tape dismounts by HPSS while we had pending requests for files available on a mounted tape. It is unclear how the prioritization of the HPSS API itself is being done and how to circumvent it (if at all possible).

There are also several infrastructure criterions which may affect performance and listed below (this is not an exhaustive list but a base reference of the main issues). However, those would affect both optimized and non-optimized restores and noted as a general reason for not being able to reach ultimate optimal restore speed.

- Delivery time: The tape may be located in different LSM, or different Solo. (a) Tape is in the same LSM, the minimum mount time: 24 sec, dismount: 24 sec or (b) tape is in a different LSM and the minimum mount time: 36 sec, dismount: 24 sec.
- Drive may need maintenance. Performance may drop when a drive becomes unreliable.
- Tape media I/O error.
- Small file causes bad performance due to an increase in the amount of tape marks (and delays associated to them). The larger the file the better.

Nonetheless, we feel and have demonstrated that a 1.21 mounts average, and regardless of the subtle features and behavior preventing to reach the asymptotic value of 1, is a large improvement over no optimization.

4. Conclusions

We have showed that tape access optimization is a crucial part of efficient use of storage systems based on a combination of tape and drives. To this extent ERADAT (formally named "BNL Batch" in early times) and the DataCarousel were developed addressing low level and high level optimization of tape access. While some of the features of the DataCarousel have migrated to ERADAT, complex user fairshare and policy handling are handled primarily at the DataCarousel level where flexibility and quick turn around testing is possible, while low level features (closer to communication with HPSS itself) are best implemented and handled in

ERADAT. The DataCarousel / ERADAT system combination have also reached a level where the tool is fully self-adapting and self-recovering from errors and external conditions.

With the combinations of those tools, BNL has achieved and demonstrated one to two order of magnitude improvements in data restore speed from tape. ERADAT was especially conceived and used for the RHIC data processing and now adopted by the LHC/US-Atlas community. The DataCarousel has been used to allow large amount of users, user groups or physics activities to share and access files and resources from archival storage in a completely transparent and fairshare manner.

ERADAT has generated interest in other communities – in 2009 and out of discussion held at HEPiX 2009 [7], it became apparent that previous discussion with colleagues at the IN2P3 had lead to the adaptation of ERADAT to the use at the CCIN2P3 center and re-branded under the name of TReqS. From then a few month of experience, similar better resource usage were observed and sharing resources between multiple experiments minimizing conflicts achieved. We view the propagation of our approach as a proof of its value.

References

[1] Brookhaven National Laboratory (BNL) – http://www.bnl.gov/

[2] The Relativistic Heavy Ion Collider (RHIC) is the first machine in the world capable of colliding ions as heavy as gold and the only machine in the world capable of colliding polarized protons beams.

[3] BNL is one of the Tier1 facilities for the LHC/Atlas project.

[4] High Performance Storage System (HPSS)

[5] P. Jakl, J. lauret et al., *Grid data access on widely distributed worker nodes using Scalla and SRM*, Journal of Physics: Conference Series **119** (2008) 072019

[6] *Efficient access to distributed data: a "many" storage element paradigm*, P. Jakl, diploma thesis 2007/2008, Czech Technical University, Praha University

[7] HEPix 2009, https://www.hepix.org/mtg/Fall%202009%20Meeting

[8] Optimizing tape Data access - http://indico.cern.ch/contributionDisplay.py?contribId=45&confId=61917

[9] ESI Handbook: Sources, Technology and Process By Adam I. Cohen, G. Edward Kalbaugh, Chapter 10 "Enterprise Storage Systems"