Efficient Access to Massive Amounts of Tape-Resident Data

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 082024 doi:10.1088/1742-6596/898/8/082024

Efficient Access to Massive Amounts of Tape-Resident Data

David Yu[1] **and Jérôme Lauret**[2]

[1] Technology Architect, Brookhaven National Laboratory
[2] Software and Computing project Leader, Brookhaven National Laboratory

Email: dyu@bnl.gov and jlauret@bnl.gov

Abstract. Randomly restoring files from tapes degrades the read performance primarily due to frequent tape mounts. The high latency and time-consuming tape mount and dismount is a major issue when accessing massive amounts of data from tape storage. BNL's mass storage system currently holds more than 80 PB of data on tapes, managed by HPSS. To restore files from HPSS, we make use of a scheduler software, called ERADAT. This scheduler system was originally based on code from Oak Ridge National Lab, developed in the early 2000s. After some major modifications and enhancements, ERADAT now provides advanced HPSS resource management, priority queuing, resource sharing, web-browser visibility of real-time staging activities and advanced real-time statistics and graphs. ERADAT is also integrated with ACSLS and HPSS for near real-time mount statistics and resource control in HPSS. ERADAT is also the interface between HPSS and other applications such as the locally developed Data Carousel, providing fair resource-sharing policies and related capabilities. ERADAT has demonstrated great performance at BNL.

1. The Role of Cold Storage in Managing the Exponential Growth of Scientific Data

As the amount of our scientific experiments data has increased rapidly, there will be a serious need to provide a long term and efficient data storage. Cold storage, such as "Tape Storage", has been an ideal solution for long term data preservation, due to various reasons like cost effective, environmental friendly and long lifespan. The tape technology has been improved in both capacity and performance, over the recent decades. Therefore the tape technology has been playing a very important role in managing the exponential growth of Scientific Data. Tape systems are great for archiving, due to the scalability and high sequential writing speed. However, randomly restoring files from massive amount of tapes usually do not get good performance because of the long latency for random accesses.

2. Effectively Using Tape Technology

The Tape Storage System at BNL[1] Scientific Data and Computing Center (SDCC)[2] has been providing mass storage services to the scientific experiments of RHIC[3] and LHC[4] (CERN, Geneva). The amount of our science experiments' data has increased rapidly, and we currently have about 100 PB of data in our tape system. We have put a great deal of thoughts into how data is saved onto tapes and how to optimize data mining and data production workflows, from a production account perspective, taking into account the time sequence and ordering of files on tape.

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. 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, a tool already introduce to palliate some of the batch short-comings. Such features included treating loading the tape with most files requests first. 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 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 code was renamed to ERADAT, standing for Efficient Retrieval and Access to Data Archived on Tape. The overall relation between the DataCarousel and ERADAT is illustrated in Figure 1.
To effectively utilizing the tape storage system, we have to manage the resource from many aspects: balance the tape drive resources for read, minimize the tape mounts, optimize the tape read, optimize the hardware resource usage and fair-share and resource handling policies.

Figure 1. ERADAT and DataCarousel relative interdependence. ERADAT sits at the lowest level, interfacing directly with the HPSS API and acts as queuing system. The facility production jobs may directly interact with it. Users or high level services typically interact with the DataCarousel.

3. ERADAT

The Efficient Retrieval and Access to Data Archived on Tape or ERADAT, is a file retrieval scheduler for IBM HPSS. ERADAT evolved from the Oak Ridge batch code as a prototype, and then modified to fit 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, T10KD and LTO drive series), and multiple staging algorithm including 'by-demand', LIFO and FIFO. ERADAT also keeps all transaction history in database, for performance reporting purposes. ERADAT collects additional data from other sources such as library controller, for cross-reference checking.

3.1. Balance the tape drive resource for read

Managing the tape drive usage for read and write is the fundamental feature for operating tape storage. ERADAT has built-in tape drive management functions.

- Allocating just enough resource for read (the rest are reserved for write). Tape drives can be allocated as dedicated resource or shared pool. Each user has a limit on resource usage.
- Allocating resources based on priority. Some users have higher priority than others, so we should reserve more tape drives for high priority users.
- Able to adjust the resources at any time, without service interruptions.

3.2. Minimize the tape mounts

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. Modern tape technologies can have very high native data transfer throughput rate; for example the transfer speed of LTO-7 media can go up to 300 MB/s, but the other latencies are making performance drop. The tape mount is the real performance bottleneck; therefore it is necessary to minimize the redundant tape mounts.

To reduce the tape re-mount rate, file requests must be aggregated by tape cartridge, so that all the requests on same tape can be read at once reduce redundant tape mounts.

3.2.1. Tape Latency Control Optimization

To provide the next level of optimization, not only all requests are sorted by tape cartridge ID but they are also sorted by file position on the tape, so that the requests can be read sequentially in order to minimize rewind and forward. Tape has quite a long latency for random accessing since the drive must rewind an average of one-third of the tape length to move from one arbitrary data block to another.

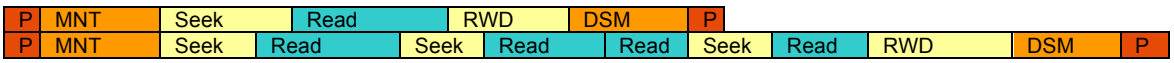

Figure 2. Tape Read Latency (top) and optimizing the read requests from the same tape (bottom).

According to the manufacturer's parameters [6], using LTO-7 as an example, we have:

- A tape delivery (cell-to-drive) time of 11 sec (P)[7]
- Mounting/Loading: 13 sec (MNT)
- Positioning on file location (first file): 59 (Average file access time)
- Data transfer (claimed): 300 MB/sec (Read), a 2 GB file should take about 7 sec.
- Average Rewinding the tape: 62 sec (RWD)
- Dismount/Unload: 22 sec (DSM)
- Place the tape back into the library slot: 11 sec (P)

In this example, the latency is 178 sec (3 minutes), but actual read-time only take 7 sec.

Aggregating files on the same tape and then read once to reduce the long latency. See Figure 2.

3.3. Optimize the tape read

The ultimate solution of optimizing tape read is to use Recommended Access Order (RAO), this is a new feature available to some new enterprise level tape drive. The RAO provides an optimized order indication to minimize the required total time period. However, the RAO feature is not available in LTO technology yet, the best way to optimize the tape read is to follow the sequence of logical position of each file on tape. ERADAT queries HPSS for detail attributes of each requested file via HPSS API call, sort the requests by Tape ID, logical position and offset.

Figure 3. When a tape is mounted, read all requested files on the same tape, sequentially.

Since large file IO is best for tape storage, small files are often aggregated via HPSS's small file aggregation feature. When reading small files from the aggregated block, offset is used for sorting the sequence within the block.

Figure 4. Reading Small files from an aggregated block.

In Figure 4, when reading small file C001, the tape is first fast-forwarded to Position 20, and then moved to offset 23, read file C001.

3.3.1. Minimize the tape positioning (rewind and forward)

Excessive Rewind and forward is a problem for tape storage as it causes extreme wear and tear on both the tape and tape drive and shortens the lifespan of both tape and drive, increases the risk of data lost. New requests may be inserted into the queue, while the queue is already in staging mode. In order to minimize the number of rewinds, the stager will continue to read the next file in the queue until the end of the queue, and then it will rewind the tape to the beginning of the first new file in the queue.

Figure 5. Minimizing the tape rewinds.

Files M, N, O are the new inserted requests, while File E is being read. The stager will continue to read O,F, G. When the last file in the queue (G) is finished, rewind to position 23, read files M and N.

3.3.2. Buffer – Optimized read performance, minimized resource consumption

There is latency between job submissions and return, a buffer may improve the performance. A buffer is a pool of threads responsible for submitting jobs, the 1st thread is the active stager, and the rest are standby. Typically, 2 threads should be enough. Figure 6 illustrated a 2 threads buffer, while File A is being staging; File B is waiting in queued. As soon as File A is finished, File B will be the next one in the line.

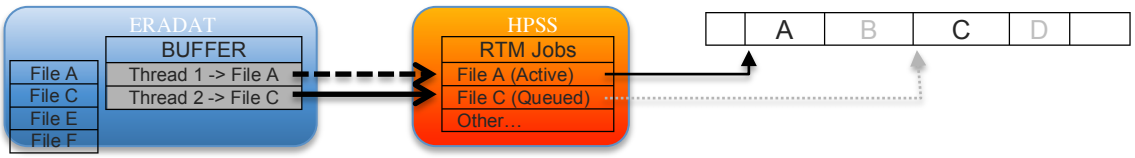

Figure 6. Buffer queue.

In Figure 7, while File A is retuned, File B should be already being staging, thread 1 should be updating the status for File A, and then submit File C. With LTO-7, 300 MB/s, we may consider to use a 3 threads model, to reduce latency. A recommendation is to not over allocate the buffer as it will not be helpful but wasting thread's resources.

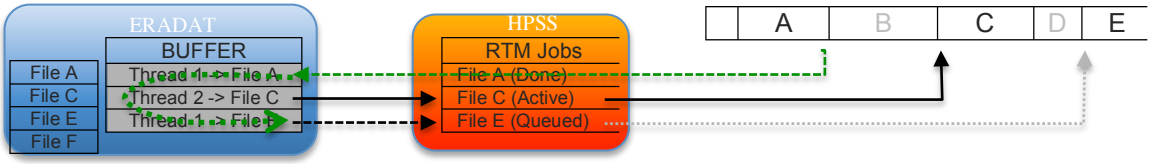

Figure 7. Buffer Queue (Return).

3.4. Optimize the resource usage

When accessing massive amount of data from wide range of tapes, increasing the bandwidth may be helpful. Tape drive sharing could be a good solution, with a good resource management.

Resource borrowing: Users sharing the same tape drives pool may borrow drive from each other.

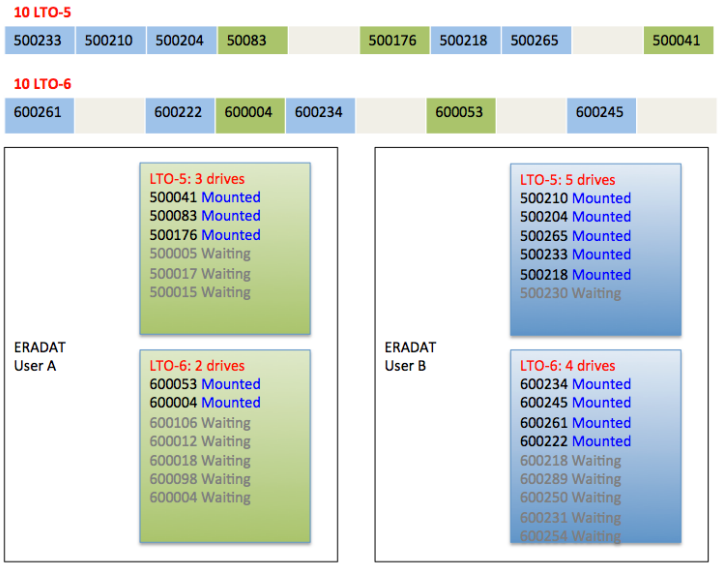

Figure 8. Resource sharing.

In the example of Figure 8, when user A needs more LTO-6 drives, user A may borrow LTO-6 drives from user B. To do this, User B needs to reduce the LTO-6 drive allocation while user A may increase the LTO-6 drive allocation. When drive allocation is changed, it will take effect immediately if there are free drives available. If all drives are busying, then it will have to wait until a drive becomes free.

Resource over-subscribing: When reading files from older generation tapes, we may use newer generation drives to increase the bandwidth. For example, we may use LTO-5 to read LTO-4 media. With this over-subscription method, we may increase the LTO-4 bandwidth immediately.

Figure 9. Drive over-subscribing.

Since LTO technology can read at least 1 generation back, HPSS will automatically try to use LTO-5 drive when no LTO-4 drive available.

4. Flexible Staging Algorithm

4.1. ERADAT staging policies

As discussed in section 3. , ERADAT implements FIFO, LIFO and "by demand" policies and those may be enabled on the administrator preference and act at a global level. However, in a multi-user environment, those basic blocks and construct policies may not be adequate to satisfy fairness. For example, consider the following scenario: user A has submitted massive requests for 100k files while user B has requested an isolated file placed on a single tape. For the sake of ultimate optimizations, a strict sorting of all files requested may be the best approach – however, user B's perception of fairness (waiting forever for his single file request to be satisfied) may not be as focused on global optimization. This is why queuing systems also introduces "shares" and fairness, allowing a balanced between optimization and expectations.

4.2. The DataCarousel

The DataCarousel (DC) is an extendable and fault tolerant policy driven framework that satisfies requests in a multi-user environment. To create the balance aforementioned above, the DC was created as a simple wrapper front end to ERADAT. Its purpose is to intercept requests from many users and act as a fairness based queuing system.

4.2.1. Relation to ERADAT and basic designs

The DC first is aware of all features of ERADTA. For example, it queries ERADAT for file placement on tape or may requests ERADAT to change policy. A usage example of the later is that in case requests comes from a single user, "by demand" policy (or strict tape ordering by biggest demand first) fulfill that single user's expectation while in a multi-user request scenario, an alteration of the FIFO policy would likely be in use. The DC can also calculate how long a request from a given user has been pending and, detecting if it is a case of "a single file requested from an isolated tape", switch the policy to LIFO (getting that requests out of the way).

The DC is essentially a collection of wrapper scripts written in perl with a MySQL back-end for keeping handling the requests. As its basic design, it respects the separation of privileges principle that is, a "thin" client inserts requests in the system and the server (a separate machine close to ERADAT and HPSS services with elevated access) is responsible for manipulating and updating the states and tracking the record progress. The server submits N records to ERADAT every few minutes up to a controllable maximum number of concurrent "batch' restore job. A note that after ERADAT restores a file on HPSS cache, a call back restores (on his behalf) the file under the user's specified location. All

failures are tracked and possibility retried. Failure varies from HPSS downtimes, drive failures but also user mistakes such as a quota exceeded. Retries can be set by the type of failure (retry count). Another feature providing resilience is that the DC also monitors the HPSS and system responses – detecting a problem, it has the ability to restart services automatically or on the contrary, throttle restores made toward specific file system. This latest feature is useful for centralized file systems (with non-scalable IO) – as many restore threads may occur simultaneously, the DC in other words handles making sure the files system where the files are restored to is not overloaded.

4.2.2. Sorting and sharing Policies

In its basic function, the DC looks at records in the order submitted and resort them according to policies. Each users may have up to a customizable maximum amount of requests in the system at a given time - when this limit is reached, other user's requests are considered first and this is done to avoid a single user starvation (no matter if a user submits 100k records, only a portion would be submitted at a time allowing a later request to "get in"). However, custom parameters may also allow super-users to have a different limit. On top of this, the DC implements sorting and sharing policies as follows.

The existing sorting policies could be NONE or TAPEID. NONE policy will sort only by the record entry time (the default). TAPEID sorting has sub-categories including a strict tape ID sorting (neither useful nor fair to users but helpful for testing) or by Time and ID (or a time slider mode). In this later "production" mode, the server sorts the records by Entry time first and tape ID second: this allows the oldest entries to be considered first while all records from the same tape those records belong to are also added to the stack of files to be restore. This may reshuffle the order by which files are restored but represents a good compromise between time expectation, fairness and optimization.

There are four sharing policies. EQUAL considers every user with request in the system to be of equal importance. They get the same share of the restores. GROUP allows putting users into virtual groups, each group having equal weights. GRPW is a composite policy allowing weighing groups: for example, group A (a production team) may have twice the amount of files restored than users in group B. A final point that the flexibility of the framework comes through allowing an administrator to write his/her own policy in a separate module (not touching the core code), loaded upon execution. This can be enabled as simply as setting the policy name in the system's control table (and having the related standalone code of the same name reside in the same directory as the core-server code).

All of those features offer a versatile set of controls allowing controlling the behavior of the system at the front-end and ensures user's expectations, fairness while preserving optimization.

5. Result

As the system is in constant production use at BNL, a massive test would be quite disruptive and slow down mission-critical operations. We therefore established an initial test to compare 704 files, 10 GB per file, out of 21 tapes, using 15 drives, a test of minimal impact on other operations.

- Direct submission: Took 270 minutes to complete. Average ~444 MB/s. Used 34 mounts.
- Using ERADAT: Took only 70 minutes to complete. Average ~1.7 GB/s. Used 21 mounts.

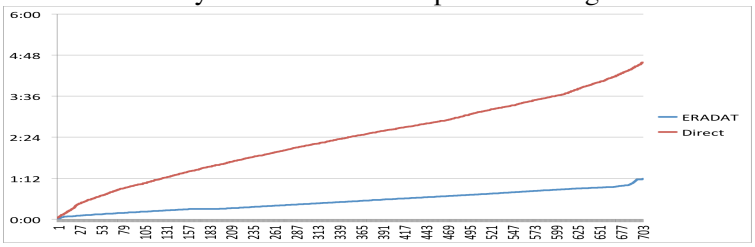

Figure 10. Direct restore VS Optimized restore.

The sub-sequent performance described in this section arises from a network congestion event and a self-inflicted non-optimal condition originating from a configuration change. It is important to understand that while a massive test would disrupt operation, in this case, we leveraged an already ongoing restore to measure a near optimal performance enhancement of our system from a recovery of a worst case scenario. In our case, a massive amount of files were requested via the DC but the limit of simultaneous requests allowed per user was increased to be in excess of 50,000 files at any time (a factor of x10 more than the maximum allowed in normal circumstances for that account and x20 more than any other users). A reasonable amount of files was showed to allow the DC to be "reactive" that is submission at pass N+ 1 would be based on the behavior of the submission pass N and possibly throttle depending on ERADAT response. This very mechanism avoids congestion as the system self-adjust to external events. In this case however, the increased limit per user for the production accounts (handling those requests) did not allow this recovery to happen and created a snowballing effect.

The results of those restores are partially seen on Figure 11 – the workflow begins with the default settings and the restore rate appeared to be around 400 files / hours. By increasing the number of tape drives from 4 to 6, we moved 1,600 files per hour for only a 50% tape before the 28th. All went well, and we decided to move the drive allocation to 8 (and boldly increase the file limit per user to 50,000 simultaneous restores for this emergency restore) and the performance peaked at 150 TB / day by September 30th, an encouraging number. Our "luck" did turn to a worst case situation as a network event aborted many transfers and this, for or a large period of time. The DC did detect and retry the failed restores as designed. But the issue was that as the network outage lasted hours, HPSS "purged" its cache and, while one of our recovery mechanism (retrying the pftp immediately and N times upon detection of failures) was foreseen as a benefit for taking care of transient issues, this had a disastrous effect in this non-transient situation. The large amount of retried failures combined with a large number of ongoing restore requests (up to 50,000 in our expanded limit) leaded to a situation where the files, previously on HPSS's disk cache, were no longer available on cache. As a result, all purged files requested at the last "pftp" stage were directly accessed from tape again without optimization (FIFO style and in no specific order). With 100ds of requests in the "retrying pftp" stage, the slowdown of mounting tapes all over again in a non-optimized manner while 50,000 requests needed to be flushed created more backlog and hence, more HPSS cache expiration. By October 3rd, and while the DC stopped further submission and sent Email notices of the condition, the ongoing performance was abysmal. We tried to allocate all available drives (16) to take care of this condition in an attempt to palliate for this incident to no avail: the performance fell to 2 TB / day and remained low for 3 days (we deliberately left this going for performance assessment and illustration purposes, making the best of a bad situation).

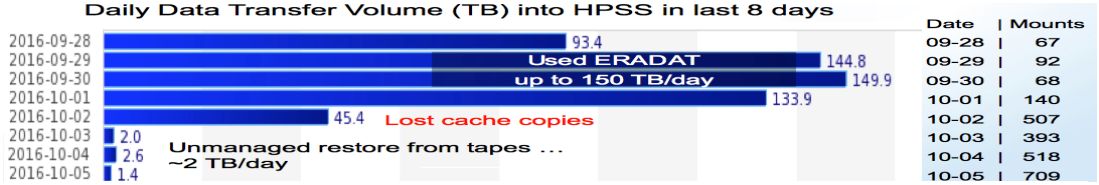

Figure 11. Restore rates with and without optimization. A catastrophic network failure caused a abrupt de-optimization starting on October 1st and causing performances to drop at a rate of 75 less as when it was optimized, demonstrating the advantage of our approach.

On October 6th, we took the step to reset all requests in a state other than "done" in the DC, re-issued a less aggressive amount of simultaneous requests (restoring to 5,000 per user) and did reset ERADAT as well (and forced an HPSS cache flush). In short, we issued the equivalent of a full system reset leaving both DC and ERADAT to act in its normal settings conditions. We saw an immediate performance increase.

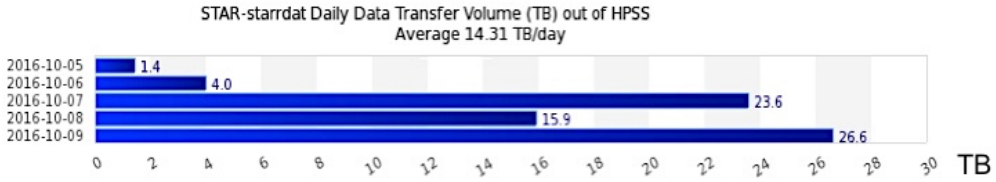

Figure 12. Restore Rate improved immediately after we restarted the system

Due to a strong state tracking in the DC system, not a single request was lost in the process. However, as our system illustrated a tangible issue, a lesson learn was also that allowing massive simultaneous restores without boundaries combined with allowing immediate retries of failed requests is not the best approach – a better approach is to not allow near-immediate retries but re-schedule the failed requests on a time cycle basis (once every N hours, all failed restores believed to be recoverable would be tried again). This option was available in the DC but was not used in this massive restore either.

6. Conclusions

In this paper, we introduced and comprehensively described two systems developed and used at BNL, providing advanced HPSS resource management, priority queuing, resource sharing, web-browser visibility of real-time staging activities and advanced real-time statistics. Those are ERADAT and the DataCarousel. Both implement policies allowing the optimization of restores by sorting or regrouping the requests. In our testing, the speed up of massive restores using both tools working together have showed restore speed 75 times faster than a non-optimized user access (using direct pftp or hsi access) as well as a nonlinear performance increase with the number of drive (+50% drive allocated to a task leads to 400% increase in file restore rate).

References

[1] Brookhaven National Laboratory (BNL) – http://www.bnl.gov/
[2] BNL Scientific Data and Computing Center (SDCC) - https://www.bnl.gov/compsci/SDCC/index.php
[3] The Relativistic Heavy Ion Collider (RHIC) - https://www.bnl.gov/RHIC/
[4] BNL is one of the Tier-1 facilities for the LHC/Atlas project
[5] Jakl, J lauret et al., 2008 Grid data access on widely distributed worker nodes using Scalla and SRM, Journal of Physics: Conference Series 119 072019
[6] Oracle StorageTek LTO Tape Drives 2016 http://www.oracle.com/us/products/servers-storage/storage/tape-storage/033631.pdf 2
[7] Average cell-to-drive time 2015 http://www.oracle.com/us/products/servers-storage/storage/tape-storage/034341.pdf 4