Table of Contents

CASTOR Tape Subsystem Twiki

Introduction

The Tape Subsystem comprises all functionality directly dealing with storage on, and management of magnetic tapes (volumes), tape servers and drives and tape libraries. This ranges from low-level tape positioning and access over disk-to-tape copying up to the management of tape drives and queues and the volume database.

Team and Contributors

The members of the development team (hosted in IT/DM/SGT) work in close collaboration with the tape operations team (IT/FIO/TSI) who are the main source for operational requirements. Also, coordination is ensured with the DM architecture team.

- **Team members**:
  - Steve Murray IT/DM/SGT (investigations; overall architecture coordination; rtcpd; VDQM2). Giulia Taurelli IT/DM/SGT (Repack2; rtcpclientd; Recall/Migration policy framework). German Cancio IT/DM/SGT (investigations; planning). Nicola Bessone IT/DM/SGT (starting Sept.08)
- (Main) **Contributors**:
  - Vlado Bahyl IT/FIO/TSI; Tim Bell IT/FIO/TSI; Charles Curran IT/FIO/TSI; Arne Wiebalk IT/FIO/TSI; Olof Barring IT/FIO/FS

Project Management

- See the dedicated Project Management page: TapeProjectManagement

Work Areas

Tape and remote copy daemons (taped/rtcpclientd/rtcpd)

- **Goals**:
  1. Acquire knowledge on tape technology: Software, Hardware, Media
  2. Understand current performance limitations in both tape write and read operations
  3. Address operational recoverability and reliability requirements and their impact on the tape data structure and format
  4. Re-engineer existing old/legacy code base using standard CASTOR development principles and foundation libraries, taking into account the above, following a low-risk migration path

Acquire tape technology knowledge

The current CASTOR tape software is outdated and not well understood by the current development team. The technology used (tape drives, SCSI devices, etc) requires in-depth technical understanding which needs to

be built up. Operational knowledge is available in IT/FIO/TSI; some development knowledge is available via A. Wiebalk which needs to be transferred to the developers in IT/DM.

Performance limitations with tape reads and writes

The current tape data formats used in CASTOR were set in place in the 1990's (ANSI AUL, NL - see link below). The NL format is not used for production data as it lacks metadata critical for media recovery (see below). There are a number of problems in terms of performance:

- **Reads**: The number of files read per tape mount is very low (around 1.5 files in average; the overhead for a tape mount ranges from 1min to 3min). This due to:
    1. related files are not written together but rather spread over available tape servers, in order to increase write performance
    2. read operations for data on the same tape are not grouped (in order to avoid stale jobs on the batch farms)
- **Writes**: The usage of migration policies enables building up file streams to be sent to tape. However, the efficiency of writing small files is low due to writing each disk file as a single tape file in the AUL file format. This format requires writing header and trailer metadata files around the contents of each data file. The following page CASTOR Tape Format shows that these metadata are separated from the contents of a data file by three tape marks; one after the header metadata, one after the data and one after the trailer metadata. The writing of tape marks is the most dominant factor in the writing of the header and trailer metadata. A total of ~5-9 seconds is taken per file to write the three tape marks and the metadata (~2-3 per tape mark). Encouragements to experiments to increase their data file size have not been fully succesful so far.

Operational recoverability and reliability requirements

- **Format**: Despite its performance disadvantages, the AUL format provides metadata information which are critical in terms of reliability and media recoverability (e.g. file name server ID, size, drive ID, etc). In order to avoid write positioning errors leading to data corruption, AUL trailer label information of the preceding file is checked when a new file is appended to the end of a tape and compared with the information kept in the database. The AUL header/trailer information also allows the identification of affected files in case of media problems (e.g. scratched/folded tape parts). Any alternative format must provide metadata information allowing for facilitating media recovery and safe positioning.

- **Stability**:
    - The software components, in particular rtcpclientd, are not robust against failures (e.g. daemon crashes). rtcpclientd is stateful and cannot be replicated (for load balancing or redundancy). After a crash, a manual database cleanup operation is needed.
    - Some of the software components, like rtcpd, are extremely delicate and critical as bugs or instabilities may (silently) corrupt production data stored into CASTOR. Also, backwards compatibility to CASTOR1 is required until CASTOR1 is completely stopped or confined. A development/deployment model is therefore required in which modules get updated/replaced incrementally, instead of a big-bang approach.

Re-engineering

In order to address the above mentioned issues, a gradual re-engineering of the tape layer components is proposed. A design document will need to be prepared with the details. The main items are shortly described here:

- **Tape format**: A new tape format will be defined in addition to the existing AUL and NL. While the AUL format structure will be kept for this format at least initially, the payload inside each AUL data

file will consist of an aggregation of multiple CASTOR files, in order to reduce the number of tape marks. The aggregation will be generated by the tape layer itself. The incoming stream (list of files) to be migrated will be aggregated to a configurable maximum total size (e.g. 10GB) and/or configurable maximum number of files (e.g. 1000 files). If a file exceeds the maximum total size it will be written in a separate aggregation consisting of that single file. On hardware with efficient tape mark handling, the number of files per aggregation can be decreased. Aggregations are not a static concept but are dynamically (re-)created on every migration; files (more precisely: segments) may be rearranged into different aggregations e.g. after repacking of a tape.

- **update November 2009**: A presentation on the benefits of new tape formats integrated in CASTOR can be found here: Tape_dev_performance.pdf

- **Aggregation format**: The aggregation format is internal to the tape layer and not visible from higher levels which still continue to see individual files (or "clusters" as defined by the stager, see below). Possible choices for the aggregation format include IL (see link below), GNU tar, CPIO, ZIP. The currently preferred choices are IL and TAR (see here for further discussion.)

- **tape gateway**: rtcpclientd will be replaced by another module ("tape gateway") which communicates with VDQM and the tape servers.

- **tape aggregator**: A new module ("tape aggregator") will be developed which will run on the tape server and which interacts with the tape gateway on one hand (via a new protocol or a database), and rtcpd/taped running on the tape server on the other hand.
  - For write operations, the tape aggregator receives the list of files to be migrated from the tape gateway and performs the aggregation (using rfio and a memory buffer) using the selected aggregation format. Each aggregation is then (asynchronously) sent over to rtcpd/taped as a single file, using the existing protocols. In case of errors during write, all files of the aggregation are flagged as failed.
  - For read operations, the complete aggregation is read out (via rtcpd) - potentially into a memory buffer - and then (asynchronously) sent over to the disk servers using rfio. Using a memory buffer to disk will be investigated, to better overcome slow disk server connections.
  - Data communication between the tape aggregator and rtcpd will be done using named pipes (over rfio) as they run on the same node. Data communication between the disk servers and the tape aggregator will use RFIO but can be replaced later by xrootd as needed.

- **rtcpd/taped**: Initially, rtcpd/taped will remain unchanged in order to preserve backwards-compatibility and to enable a low-risk migration path.
  - In the short term, small changes may be required to accomodate the existence of a new tape format.
  - In the future, the tape aggregator will be extended to cover their functionality (after stoppage of CASTOR1). This will also allow for enhancing functionality such as direct file positioning within an aggregation, for avoiding always having to read complete aggregations.

- **name server**: The name server needs to be enhanced to allow storing/retrieving the mapping between segments and the aggregation they belong to (and the position within a given aggregation).

Discussion points

- **"Aggregations" vs. "Clusters"**: Tape level aggregations as proposed here are an independent, lower-level concept than "clustering" as proposed by the stager working group, but both concepts are compatible. The following should be done on the stager side to further increase efficiency (but there is no functional dependency from the new tape subsystem):
  - Migration policies should group files belonging to the same cluster to be written to tape within the same migration stream(s). This will ensure that files belonging to the same cluster are stored together in aggregations.

- The recall policies should group as many files from the same cluster and/or aggregation, possibly by extending user requests for individual files to cover other files in the aggregation ("speculative read ahead", [[FIOgroup.SpeculativeRecalls]link]). The mapping of file to aggregation can be resolved via the name server.
- The potential benefits of having FTS sending aggregations also has been discussed. Here, the network is less the bottleneck but rather the achievable disk throughput; therefore FTS uses parallel connections for multiple files and fills up the available bandwidth, which fits naturally with the file spreading done by CASTOR. From a throughput perspective, serialization of files would therefore not be an advantage (nor a disadvantage, in case the bandwidth is filled up anyway).

- **Sequencing within aggregations**: There is no sequencing worth to be respected by the tape layer as the sequence of user operations (e.g. file creation) is 'lost' by the stager (due to scheduling/load balancing across disk servers, application of policies). The sequence of files within an aggregation is therefore meaningless and cannot be exploited.

- **Dedicated disk cache layer**: Efficient tape recalls/migrations require high bandwith availability between the tape server and the disk server sending/receiving the data. With the speed of modern drives (which nominally can reach 160MB/s) the 1Gb network link (~110MB/s) of a disk server will get saturated. This implies that during migration/recalls, other data accesses (e.g. end users) should be avoided on a particular disk server which means that all other data on that disk server becomes unavailable during the time tape data transfer is happening. With increased disk server sizes (10-15 TB) this becomes a significant limitation. A dedicated disk layer would allow to decouple user I/O access from the I/O load generated by tape drives running at maximum speed. The following options are envisageable:
  1. Define a standard Castor disk pool (no user access) where files get replicated on creation (via "replicate on close"). Migrations/recalls will be run from that disk pool. The configuration of that disk pool requires investigation (many small batch-node like servers allowing for high and fast throughput from/to tape? Large disk servers with higher retention and longer streams?)
  2. Use the local tape server disk for data collection and caching. This would require local disk scheduling on the tape servers; also the attached tape drive is idle during the time data collection / caching is running.
  3. Complementary to 1. or 2.: create the aggregation outside the tape layer. Tests have demonstrated that there is little overhead resulting from the concatenation of small files scattered across multiple disk servers using rtcpd, so there is no advantage in performing this step outside the tape layer.

- **Hardware support and evolution**:
  - Tier-1 sites. What is the lowest common denominator? Will block positioning support (needed for positioning within aggregations) work on tape drives used by Tier-1's?
  - Impact of FC fabrics

Roadmap

A coarse-grained roadmap is listed below (last updated 29/6/09 - **draft**). Note that there are dependencies on the 2.1.9 series of CASTOR, in particular for the new threading library provided in the 2.1.9 core libs. Also, there are DB schema changes (mostly additions) but also some modifications in the associated objects e.g. tape copies. Having the software running on 2.1.8 would require isolating the dependencies and backporting as appropriate. This has not yet been addressed.

First phase: provide the new gateway and aggregator (working in bridge mode - only AUL/NL tape format support)

name	description	duration	date
Complete protocol specification	Finish the specification of the (connection less) transfer protocols between the new gateway, aggregator, and VDQM, as well as between the aggregator and rtcpd.	1w	
gateway DB schema	Finalize the tape gateway database schema (requires discussion with stager team)	1w	
development of gateway+aggregator (in bridge mode)	complete the developments of the first version of the tape gateway and aggregator for bridge-only functionality (no aggregations)	4w	
development of tpcp recall/migrations	complete the developments of tpcp for recalls and migrations	2w	
development of tpcp dump	complete the development of tpcp and aggregator for dumps	1w	
documentation	Write documentation for gateway/aggregator protocol, aggregator/rtcpd protocol, tape gateway DB schema, workflow, and user documentation	2w	
test plans	develop test cases for aggregator bridge functionality, including upgrade/downgrade to/from current rtcpclientd	1w	
packaging	provide installation + init.d scripts, man pages, example config files	1w	
test phase	ensure recalls and migrations work correctly, verifying in particular the drive allocation and end of transfer protocols, and boundary conditions + error recovery test cases	3w	
release	release of gateway/aggregator first phase		W35 == end of August 09

The software delivered in the first phase aims to provide backward compatibility:

- The aggregator will be installed in addition to rtcpd on the tape servers. The aggregator does not access directly the tape device but does it via rtcpd. Stagers still running rtcpclientd will continue to directly communicate to rtcpd.
- Files written to tape by the tape gateway+aggregator can be read by rtcpclientd+rtcpd.

Second phase: provide full support for aggregations (ALB block format) in the aggregator

name	description	duration	date
development of tpcp verify AUL	complete the development of tpcp+aggregator for verification, AUL format	2w	
development of tpcp verify ALB	complete the development of tpcp+aggregator for verification, ALB format	1w	
development of aggregator(full support for aggregations)	Design and develop the aggregation building part inside aggregator, including modifications required at the level of rtcpd, and tape gateway	4w	
block positioning	reverse engineering of current positioning logic for recalls (mostly taped/posovl), modify to support block positioning within, and not only at the beginning of, an AUL-embedded aggregation. If this is not possible, aggregator needs to read out the whole aggregation, which is less efficient	3w	
tape file packer/unpacker	complete the tape file packing and unpacking CLI's to work with tpread/tpwrite	1w	
packaging + doc		1w	

	Update documentation for aggregations and tape file packer/unpacker. Provide installation + init.d scripts, man pages, example config files		
test phase	ensure aggregations can be read and written correctly	3w	
release	release of gateway/aggregator second phase		W48 == end Nov 09

As data written to tape by a stager should be readable by any other stager, a careful deployment should be envisaged; a deployment plan needs to be worked out with the stager and tape operation teams.

- Prior to writing any data in ALB format, the new aggregator with ALB support should be deployed on all tape servers, and all stagers have to be upgraded from rtcpclientd to the tape gateway.
- Enabling writing in ALB format should be done progressively - first by running field tests with tapes using fake data, then with tapes to be repacked (where source and destination files should be verified), before enabling ALB by default on any stager

Meetings and Presentations

- Meeting with FIO/TSI, 1/7/08 link
- Presentation at DM arch meeting, 16/7/08 link
- Meeting with FIO/TSI, 25/7/08 link
- Presentation at HEPiX, 18/10/08 link
- Poster at CHEP 2009 (Prague), 26/3/09 link
    - The corresponding draft paper submitted can be found here: chep2009tape.pdf: CHEP 2009 paper
- Presentation on testing status at the Castor F2F meeting, 12/10/2009: link
- Measurements of tape (migration) performance using different tape formats, November 2009: Tape_dev_performance.pdf
- Presentation on XROOT support for RTCPD: rtcpd.pdf:
- Presentation on modifications to RTCPD for the new tape format: alb.pdf:

Links

- Tape Gateway design documentation
- Tape reading, debugging and verification
- Test battery
- Aggregation formats
- IL metadata format proposal
- Tape labels at CERN: overview, specification, current implementation
- RTCOPY documentation: design proposal
- Speculative recalls analysis (link)
- Savannah bugs on CASTOR tape subsystem

-- GermanCancio - 11 Jun 2009

This topic: DataManagement > CastorTapeSubsystem
Topic revision: r31 - 2010-08-12 - EricCano