Overview of LHC Storage Operations at CERN: CERN IT file-based physics data storage operations in 2011/2012

Overview of LHC Storage Operations at CERN

CERN IT file-based physics data storage operations in 2011/2012

J.Iven, M.Lamanna; CERN IT DSS-FDO

Introduction

The main responsibility of CERN in the LHC Computing Grid (LCG) is to receive and store the RAW data, to make them available for quasi-online processing and for further distribution to selected computer centres and provide long-term storage (tape recording). Derived data generated at CERN can also be redistributed together with simulation samples. This is described in details for each experiment workflow in the LCG Technical Design Report [LCG-TDR]. In addition to this critical functionality CERN provides capacity for final user analysis, both as a Grid centre and as a traditional computing facility (interactive and batch access).
CERN-IT operates two large-scale disk-based file stores:

- CASTOR – in production for many years – now handles the Tier0 activities (including WAN data distribution) as well as all tape-backed data;
- EOS – in production since beginning of 2011 - is the CERN strategic choice to address the fast-growing need for high-performance low-latency data access for (disk-based) user analysis

In 2011, EOS had been validated in close connection with the LHC experiments for several months under production conditions. EOS is absorbing a large fraction of the disk-only service originally provided by CASTOR. In about 12 months the original 1 PB testbed grew to around two 6 PB installations for ATLAS and CMS.

In parallel, CASTOR evolved with major improvements targeting stability and maintainability. The internal scheduling of requests has been rewritten; the database structure and the tape subsystem have been simplified and made more robust and efficient.

This paper compares the two systems from an operational perspective (setup, day-by-day user support, upgrades, resilience to common failures) while taking into account their different scope. For both systems, the respective upcoming changes will be discussed.

CASTOR overview

CASTOR-2[1], the "CERN Advanced STORage manager" ([CASTOR]), is the successor of the SHIFT storage architecture in use at CERN since the early 90's and the subsequent CASTOR system. It is a full-fledged hierarchical storage manager (HSM, i.e. automatically writes files to tape, recalls them on demand from a tape library; it is capable of fully managing the disk space in front of the tape system) and covers a wide range of use cases:

- the classic tape-backed storage (D0T1), where a relatively small disk pool serves as a low-latency cache automatically managed by the system;
- disk-only (D1T0) and disk-and-tape (D1T1) storage, where the user/experiment controls the cache lifetime, files are also backed up on tape in the second case;
- disk-only temporary storage: files are kept only on disk. Once space runs low they are cleared out automatically according to standard policies such as "least recently used".

At CERN, CASTOR has a unified namespace for all stored files and uses a common tape infrastructure. Each of the 4 main LHC experiments has a private disk-layer instance and there is a public instance for all the other user communities. Additional instances exist for internal services such as testing and tape repacking. Each instance is split into several disk pools (typically in the range between 100 TB to several PB) to handle different types of data with different access patterns. Each instance has tape-backed up and disk-only pools.
Namespace and actual storage are orthogonal: a given file can be present at the same time on one or several disk pools and/or on tape.
All disk files are stored in RAID (almost entirely RAID-1) with the dual goal to hide single-disk failures and to provide reasonable durability for disk-only pools.

CASTOR has several advanced features, such as multiple file replicas on tape and/or within a disk pool (beyond the low-level RAID redundancy), and several data and access protocols (the original RFIO access has been supplemented by other popular protocols like SRM, gridftp and xrootd).

1 referred to simply as "CASTOR" in this article

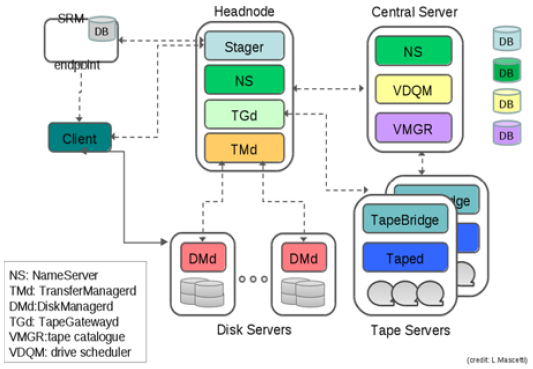


Figure 1: (simplified) CASTOR architecture

The CASTOR architecture (Fig. 1) is based around a set of database services that provide consistent state for higher-level functionality. This ensures excellent "horizontal" scalability since more processing capacity can be added to central services while preserving consistency via the shared database information. The code and development cycle are mature, with a goal of two releases per year (one for deployment during the winter shutdown, one aimed at the technical stop in late summer).
Other institutes operate CASTOR, notably two WLCG Tier-1 centres (RAL and ASGC).

Issues

CASTOR is a complex system to operate and use. Currently the operational effort is spread across two teams for disk and tape. Operations rely on the CERN-IT database services and on the CERN user support and system administration infrastructure. The complexity of the operations requires specialist knowledge; this limits the (desirable) sharing of personnel across all the services provided by our group.
In parallel, user support and troubleshooting consume non-negligible effort due to the system complexity. CASTOR handles files using an intuitive name space but the presence of different pools (different characteristics, different access rules, automatic replications across pools) tends to confuse also relatively expert users.
The original HSM design is apparent in several areas, for example in reaction to exceptions (e.g. clients disappearing without explicitly closing output files): CASTOR tries to keep such data, which creates operational problems both on the user and on the operations sides ("dark data").
The original design was strongly influenced by disk space being a scarce resource. It favoured optimised tape writing over serving data to clients at high concurrency. The availability of large disk farms (and the evolution of analysis tools and experiment data structures) has opened up opportunities for new workflows. CASTOR provides solutions for these, but these were not its original focus.
CASTOR copes effectively with its original design goals and its critical role for the LHC data acquisition but it is less suited to the random fluctuations of massive user analysis.
A sore point for user analysis is that CASTOR lacks management features that would allow to effectively implement space quotas or to delegate all access control to experiment administrators; it also has an insufficient security model.

Recent changes in CASTOR

TransferManager: On the disk layer, a major improvement was the replacement of the I/O scheduling component. Previously this had been based on a commercial batch scheduler (Platform LSF) with a CASTOR-specific plug-in (LSF had originally been chosen because of its flexibility, but very little of that was actually used in CASTOR). This solution was both complex to manage (e.g. new disk servers needed to be registered explicitly) and the cause of several incidents. In particular, it limited the rate of file read/write requests to about 20 Hz (a rate which had been approached several times in 2011), introduced scheduling delays and caused general instability under high load.

The solution was to replace LSF by a CASTOR-specific greedy "random allocation" scheduler (*TransferManager*). This has proven to scale well, both on the request rate (>200Hz; capped in production to 75Hz in order to prevent overload of other components) and queue size. The scheduler itself is largely stateless. Originally deployed in 2011 as a configurable option, the *TransferManager* quickly proved a superior solution and since early 2012 (version 2.1.12) is the only CASTOR scheduler.

TapeGateway: On the tape side, significant improvements (factor 10) in tape write speed were achieved in 2011 via the progressive introduction of buffered tapemarks, stricter tape access policies (to forcefully group tape operations and use efficient bulk access) as well as a partial rewrite of the CASTOR tape subsystem (*TapeGateway*) – see [CASTORTAPE1] for more details. In addition, the CASTOR tape subsystem now supports more hardware types (IBM TS1140, Oracle T10000c, LTO-5), and tape content is continuously verified – both for tapes that have not been accessed recently (so-called "dusty" tapes) and for freshly filled tapes ([CASTORTAPE2]).

Both changes together allowed to reduce the CASTOR code base by about 70k lines (10%).

Changed experiment behaviour

Nowadays CERN has installed several PB of disk space for physics data per experiment. This created the opportunity for concurrent, fast-turnaround analysis over larger and richer data samples by growing user communities.

However, some of the data handling tools and protocols had been designed and optimised during the LHC preparation phase, with a lot of emphasis on (well-structured) production activities. As in any field, the emergence of new access patterns brings a need for modifications and sometimes of new approaches.

This is true across the entire LHC computing area but particularly apparent in the activities with a genuine distributed character (FTS, SRM, data federations; batch computing where efficiency via data locality collides with opportunistic usage of CPU and storage resources).

In the specific case of CASTOR the coexistence of radically different activities (production and analysis) puts additional pressure on the system and occasionally causes negative interferences. Solving these through sophisticated features in the software or deployment introduces additional complexity and will ultimately be the origin of (new) problems.

EOS

During 2010, the CERN IT DSS group decided to address these issues by moving the analysis use case into a separate disk-only system - [EOS]. Its design goals were:

- low-latency at high concurrency both for metadata operations and actual data access
- service resilience in case of overloads and failures
  - Tuneable data replicated across different hosts (file base or block based replication)
  - Self-balancing and self healing in response to common failures (disk failures, and individual machine outages)
- Robust client behaviour (inherited from the xrootd protocol) to cope with temporary unavailability due to overloads, failures or maintenance operations

The EOS [2] architecture (Fig. 2) has been built on the existing XROOT/Scalla storage system ([XROOTD]) extended via a very fast (in-memory) persistent namespace, a message queue and replica management. A major difference with CASTOR is that the user I/O is not scheduled, instead the system will back off once individual disk servers are overloaded.

Since data is replicated across machines ("RAIN" model), no redundancy within a machine is required (JBOD can be used instead of the RAID setup on CASTOR).

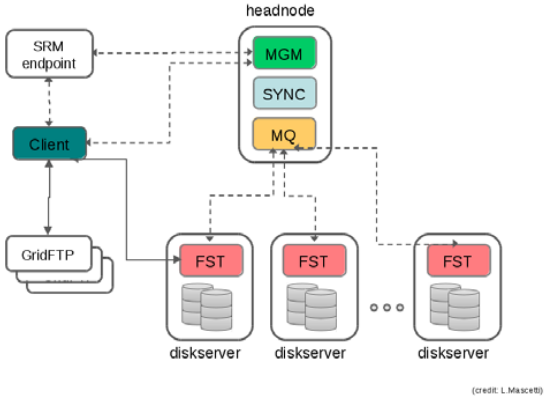


Figure 2: EOS architecture

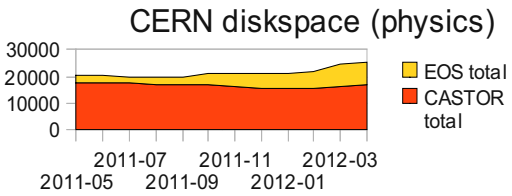
The new system got prototyped together with ATLAS ([EOSATLAS2010]) and transitioned into full production mode at CERN for both ATLAS and CMS in May 2011 ([EOSATLAS2011]). A smaller EOS instance runs at FNAL.

The EOS implementation allowed to have a fresh look to the areas where CASTOR was weak. As an example, since EOS supports robust quotas, the split of resources into separated disk pools (as on CASTOR) is no longer needed; this permits to efficiently share disk and I/O resources across an entire EOS instance.

The primary data protocol on EOS is xroot[3]. "Grid"-style access via SRM and gridftp is provided via dedicated gateways, but these are less performant than their CASTOR counterparts.

Current storage usage

At the time of this writing (2Q2012), CASTOR was still the larger of the two storage systems, even when just looking at its disk layer - it had twice as much capacity (17 PB vs. 8.5 PB), and three times as many machines (1280 vs. 400 [4]).



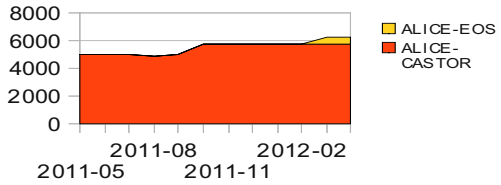
2 This is not really an acronym – several such have been proposed after the name stuck, such as "exploration of storage" or "exabyte online storage"

3 WLCG considers to use this for federated storage; CASTOR is moving to this as well

4 EOS machines tend to be more recent models, i.e. higher capacity

Those experiments using EOS have agreed that essentially all of the CERN-WLCG 2012 additional disk capacity (about 15 PB) will end up in EOS, and the move of disk-only activities from CASTOR to EOS will continue.

ALICE storage usage



The ALICE CASTOR disk layout is very efficient, with two pools of comparable sizes – one for tape-related activity and another pool for disk-only access. EOSALICE has been added beginning of 2012, but since ALICE is already using the xrootd protocol exclusively, even for wide-area access, so we expect the migration to be relatively simple. The plan is to arrive to a 50%-50% split of resources by autumn 2012 – essentially one tape-backed up CASTOR pool for tape related activities and EOS for the activities now hosted on the disk-only CASTOR pool.

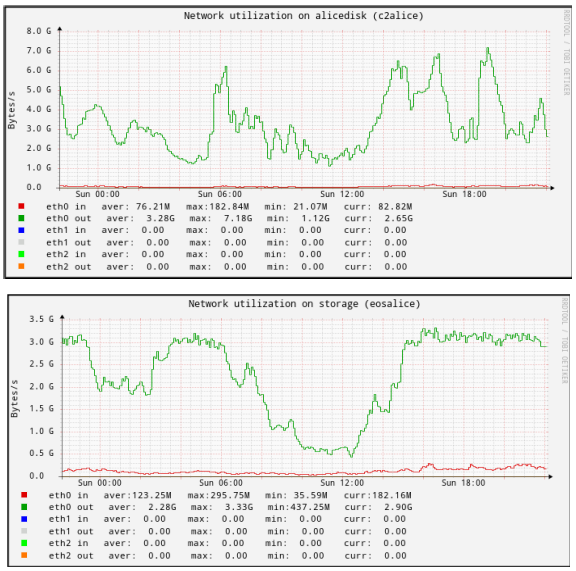
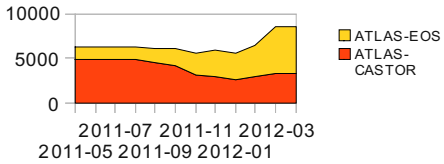


Figure 3 - bandwidth usage

Although it is a recent addition, EOSALICE is already seeing intense usage from analysis jobs - despite being only a tenth in size of CASTORALICE/ALICEDISK, it delivers already 60% as much data (Fig. 3 shows a snapshot).

ATLAS storage usage

ATLAS was the first experiment to rely on EOS in production and used the migration for a major cleanup on the CASTORATLAS instance - 5 CASTOR pools were switched off and the corresponding activity (and disk resources) went into EOSATLAS. Three more pools are scheduled for removal. EOSATLAS also absorbed the CASTORCERNT3 instance (a dedicated CASTOR instance shared between ATLAS and CMS; decommissioned at the beginning of 2011).

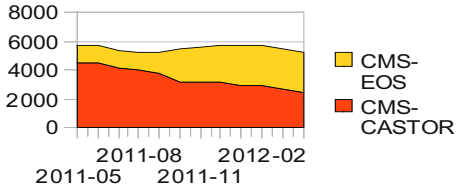


The EOS migration was facilitated by a previous policy change by ATLAS on CASTOR. ATLAS separated tape file access from user activity already in CASTOR: "normal" users no longer could request files from tape (instead, such requests were handled by the ATLAS DDM tools [DQ2] and to a certain extent by an experiment manager).

Lastly, ATLAS spent considerable effort in trying to use efficient local (per-site) access protocols and in particular they reduced their use of SRM. For CERN-internal access, the ATLAS production and analysis system ([PanDA]) now uses the xrootd protocol with direct data access. On the other hand wide-area transfers through DQ2 use direct gridftp transfers on EOSATLAS instead of going via SRM.

CMS storage usage

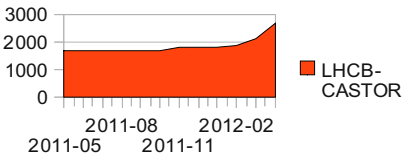
Similar to ATLAS, CMS moved several existing CASTOR pools (7 out of 15) into their EOS instance, and (as ATLAS) went away from the dedicated CERN CASTOR analysis instance (CASTOR CERNT3).



On EOS, CMS uses the new quota system extensively and has integrated the EOSCMS user management directly into their account management tools.

For data transfers, also CMS prefers to use the xrootd protocol, including for some types of wide-area transfers via their transfer manager [PhEDEx].

LHCb storage

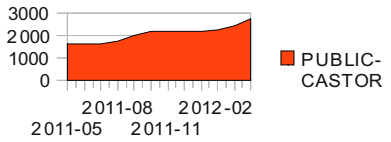


LHCb has undergone a major CASTOR pool consolidation

in 2011 - now only 5 pools are left, with most of the resources in *lhcbdisk* and *lhcbtape*. (disk-only and tape-backed, respectively). LHCb also intends to continue using SRM/gridftp as preferred wide-area transfer protocol.

LHCb is starting an evaluation of EOS. The detailed plans and time scale for a future migration will depend on their findings.

non-LHC experiment storage



At CERN, the public CASTOR instance serves the non-LHC experiments (active fixed-target experiments such as COMPASS and NA62, activities like the AMS mission and ILC studies or completed experiments like the LEP ones). Usage is rather varied, from full grid-style access via SRM (as exercised by ILC) to largely local access (or sometimes pure archival use). The CASTORPUBLIC instance is rather fragmented (16 pools, the smallest with just 26 TB), since most experiments want to decouple at least data taking from analysis. Experiments and activities without dedicated resources can use shared pools like DEFAULT (user analysis) or CDR (data taking activities for low-rate experiments).

A "shared" EOS instance is under discussion since several experiments have showed interest, but would need resource commitments (storage space, migration effort) from a few of the larger experiments in order to become viable.

All experiments on CASTORPUBLIC are being encouraged to investigate using the xrootd protocol for data access both for a possible EOS migration but also to prepare for the expected RFIO deprecation in CASTOR.

Operational comparison

Running two similar storage systems side-by-side gives the opportunity to compare them. However, the different target workflows need to be taken into account (e.g. for the experiments raw data recording on CASTOR has a higher priority than EOS access for user analysis). In addition, the CASTOR disk subsystem has the additional complexity of the tape infrastructure that cannot always be factored out in order to obtain a pure "disk"-layer comparison.

Hardware

Both EOS and CASTOR use the same basic style of disk storage ("disk servers" - whitebox rack-mounted machines with typically between 16 and 36 3.5" disks). For chronological reasons EOS benefits from using more recent hardware models, but this advantage will go away after a few hardware generations. The per-box hardware cost and power consumption between the two systems over longer periods is identical.

Only recently models without a hardware RAID controller have been bought and integrated into EOS (the JBOD setup in EOS does not require such controllers). The plan is to continue using hardware RAID on CASTOR. The hardware cost difference is expected to be small assuming CASTOR RAID1 and EOS 2-replica setting (both are the default and by far the dominant configurations).

Both EOS and CASTOR have a certain cost overhead through hardware that does not immediately contribute to available data storage space, namely nodes with a specific role (such as headnodes or SRM servers) as well as test/development infrastructures and machines in maintenance.

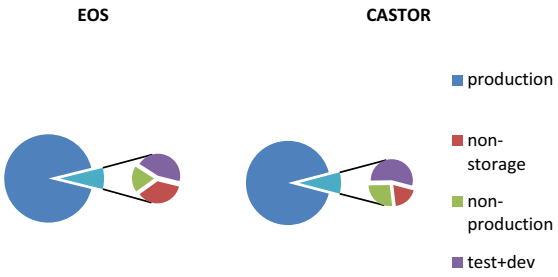


Figure 4: HW cost efficiency and overhead breakdown

Overall efficiency is comparable (and high) for the two systems, in particular measured by cost - the non-storage systems and services are typically on cheaper hardware (or even virtual machines) than disk servers.

Not surprisingly, the amount of low-level hardware errors on the two systems is similar (at about 0.5 exceptions / machine / month). Hardware failures tend to be dominated by failing disks which is largely independent of the top-level software. On CASTOR these disk errors are "hidden" behind the RAID controller (dealt by the system administration team without direct involvement of the CASTOR operations). EOS machines see them at operating system level, where they usually trigger various sensors at once (dead disks give SCSI-level I/O errors, controllers may hang while trying to reset that disk, eventually the filesystem will go read-only and/or get unmounted etc). Some effort had been spent on EOS in order to group and correlate such errors again - Fig.5

shows a spike where this correlation was not successful.

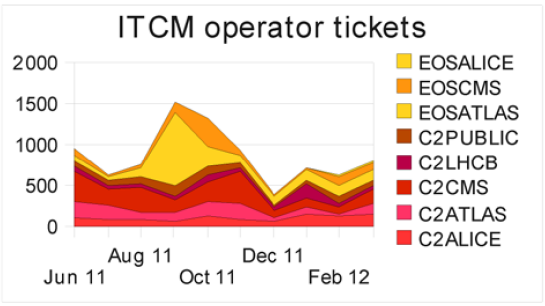


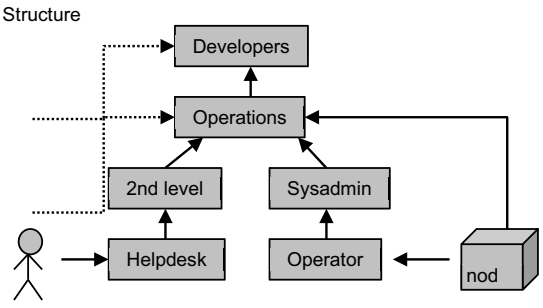
Figure 5: absolute number of alarms

Both systems rely on CERN computer centre tools ([QUATTOR]). EOS as the newer system is better integrated with these (CASTOR has some legacy workflows). Some particularly complex and/or fragile components have been rewritten for EOS and they are now being reused in CASTOR.

User Support

The basic support structure for both storage systems is identical. Each user request goes though a CERN-wide ticketing system and is triaged across the various support teams. The escalation procedures include the developers whenever appropriate (although typically this would go via bug report in a different tracking system).

Machine-level issues follow a similar route via the 24/7 CERN-IT operators and a dedicated team of service-independent system administrators. In this phase the experiments prefer to use their internal support teams for most issues on EOS - as such the ticket numbers do not reflect actual overall support effort.



Tickets raised on behalf of the experiments via the WLCG support tools (GGUS) also go through the support infrastructure, but in parallel are sent directly to operations level (since these tickets typically have a higher weight). For serious incidents, a call-out can be requested by a number of "power users" (typically the experiments' data taking shift staff). Both CASTOR and EOS rely on "best-effort" after-hours availability. Since CASTOR outages may compromise the ability to store raw data, CASTOR tickets tend to be raised at higher urgency - all of the GGUS ALARM tickets received over last year where for CASTOR.

The effort for supporting both EOS and the CASTOR disk layers (EOS: ~1.5 FTE, CASTOR ~3.3 FTE in 2011) is roughly comparable when taking the respective capacities into account. However, this does not include several other supporting services (such as database administrators, relevant only for CASTOR).

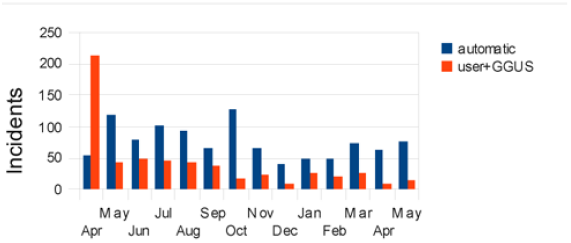


Figure 6 : CASTOR tickets (all instances/support lines)

As can be seen from Fig.6 the (CASTOR) tickets in 2011 are dominated by automatically-generated tickets, they are created as a reaction to certain hardware issues (e.g. reboot of a machine in a critical pool for firmware upgrades). Since autumn the number of new CASTOR user+GGUS issues has decreased, this decreased workload has been noted as well by the support team[5].

Operations

Looking at the effort in order to "keep the systems running", a series of basic routine task emerge:

- **Software updates:** both EOS and CASTOR are packaged in RPM format. They are configured and deployed via Quattor. A CASTOR updates typically need additional database-side operations (in most of the cases these do not affect the upgrade downtime, but have sometimes generated incidents or extended downtime).
- **Troubleshooting:** CASTOR has a dedicated (syslog- and database-driven) logging system with web interface (DLF) which has been useful in putting together the various pieces for a single transfer. DLF also provides aggregate monitoring information, but has turned out to be rather fragile. It is currently being rewritten. In contrast, EOS just uses plain text log files. Since there are fewer machines involved in a transaction, the lack of a dedicated debug tool is not yet a major concern, and the reduced number of "moving parts" (daemons) makes error analysis often more straightforward. Nevertheless, the data volume (30 GB/day) means that other log analysis solutions (such as [SPLUNK]) are being investigated for EOS.
- **Draining:** both CASTOR and EOS need to drain

5 The request spike in Apr 2011 is due to the introduction of a new ticketing system

disk servers, either in preparation for a potentially-destructive intervention or after a serious error has been detected putting data at risk. EOS drains automatically as soon as a disk error is seen, whereas CASTOR relies on the RAID controller to cover single disk errors - as such drain operations are a core part of EOS and hence much better supported than their counterparts on CASTOR. In addition, CASTOR has to take the data out of a single machine, where EOS will use all available replicas during draining, and so is not limited by the bandwidth a single machine can provide. Nevertheless, both CASTOR and EOS regularly will leave residual data files after a drain operation (e.g. dark data, inconsistent checksums etc), and these need to be handled manually on both systems.

- **Balancing:** the relative fill grades of individual disks and nodes can vary over time - new hardware arrives empty, and both CASTOR and EOS allow incremental writes without pre-announced file size, so can therefore fill up a file system beyond what would usually be permitted. The difference is that EOS will actively rebalance the fill grade by shuffling files, whereas CASTOR relies purely on user-triggered deletions (and garbage collection in case of tape-backed pools) in order to correct the situation, and little-used pools need manual intervention. EOS also tries to minimize such cases by preallocating space for all situations were the final file size is known in advance (whole file copy over *xrdcp* or *gridftp*), CASTOR relies solely on a (tuneable) estimate.
- **Consistency:** the two systems support preset checksums (adler32), will store checksums next to the actual data, and use them consistently across all data access operations, including internal ones. CASTOR has an optional background data file scrubbing process and runs an independent disk scrubber ([FSPROBE]), EOS has a built-in disk scrubber and triggers regular data movements via balancing. The difference is that CASTOR always needs manual action for corrupted files (since usually the only existing copy is affected); EOS can fall back to the secondary copy unless the corruption is due to a logical error.

Overall, EOS appears to be easier to operate than CASTOR. The difference is due to the simpler design but EOS as the newer system has of course also benefited from the operational experience on CASTOR.

Upcoming changes

Both storage systems are still under active development. At this point, CASTOR is not supposed to get new functionality, but increased efficiency (or the need to adapt to external conditions, such as new/faster hardware) still warrants changes. EOS - as a rather recent system - is still evolving more and its feature set is not yet complete.

CASTOR

The CASTOR tape subsystem rewrite is still ongoing, after the changes done on tape migration now the recall part is being replaced to also use more efficient bulk protocols. In the intermediate term, work is foreseen both on the metadata protocols (namespace and stager), as well as changing all data access (including internal tape/disk communication) to use the xrootd protocol.

CASTOR code and deployment will be further streamlined to reduce unnecessary complexity, but the current feature set will stay supported.

EOS

EOS has several areas were improvements are planned in the short- and medium term:

1. the consistency between namespace/metadata and actual content on disk needs to be regularly verified. The current EOS "FSCK" operation runs on the MGM, but is a rather heavy operation and therefore runs only at fixed intervals. It will be replaced by continuous background checks, to a large degree delegated to the individual diskservers (with a special attention to "orphaned" files). At the same time, automatic recovery actions will be extended to repair more of these inconsistencies.
2. While the in-memory namespace (built around Google hashtables) is very fast, it imposes scalability limits - essentially the RAM of a single machine. Current memory layout is not much optimized yet, so a 48 GB machine will hold about 35M files. Alternatives will be investigated, from converting the pure in-memory solution to a file-backed storage with in-memory cache for hot data, to using a NoSQL database.
3. The current high-availability solution for the MGM (warm standby machine with continuously-synced on-disk copy of the namespace) requires an external trigger to initiate a failover, and then may take several minutes change a DNS alias and boot up the namespace. An interesting alternative would be load-sharing readonly slaves with an internal election algorithm.
4. The only redundancy method currently in production is file-based, with a per-directory tuneable number of replicas. Several block-based algorithms have been implemented in EOS but need deployment. The decision whether a file

should have full physical replicas or the more space-efficient (but slower to read write) block-based replicas should be ideally taken by EOS itself, based on access patterns.

As such, EOS is in full production but certainly has a few interesting releases ahead of it.

Summary and outlook

The two large-scale file-based storage systems at CERN are both fully operational and intended to be complementing each other. EOS already delivers on its promise of simplified operation while reducing the support load and complexity on the CASTOR side, but both systems still have margins for improvement. Together they are well positioned to receive LHC data for the medium-term future. Longer-term, the various "cloud" storage services promise to be unbeatable on price, given their economies of scale, but to successfully utilize these major investments from the experiments and user communities will be required.

Of course, the success of CASTOR and EOS depends on many contributions - our thanks go to our colleagues in DSS and the experiments, the CERN-IT/DB CASTOR database administrators, the system administrator team and the CERN helpdesk.

References

LCG-TDR: The LCG TDR Editorial Board, LHC Computing Grid – Technical Design Report, 2005, http://lcg.web.cern.ch/LCG/TDR/LCG_TDR_v1_04.pdf

CASTOR: CASTOR development team, http://cern.ch/castor/docs.htm

CASTORTAPE1: S Murray, Tape write efficiency improvements in CASTOR, 2012

CASTORTAPE2: G Cancio Melia, Tape status and strategy at CERN, 2012

EOS: A J Peters, Exabyte Scale Storage at CERN, 2010

XROOTD: XRootD/Scalla documentation, http://xrootd.org/docs.html

EOSATLAS2010: Large Scale Eos Data Management and Performance Test (LST2010), 2010, https://twiki.cern.ch/twiki/bin/view/Main/LST2010

EOSATLAS2011: S Campana, D C van der Ster, A Di Girolamo, A J Peters, D Duellmann, M Coelho Dos Santos, J Iven and T Bell, Commissioning of a CERN Production and Analysis Facility Based on xrootd, 2011

PanDA: T Maeno, PanDA: distributed production and distributed analysis system for ATLAS, 2008

DQ2: M Branco, D Cameron, B Gaidioz, V Garonne, B Koblitz, M Lassnig, R Rocha, P Salgado and T Wenaus, Managing ATLAS data on a petabyte-scale with DQ2, 2008

PhEDEx: L Tuura, B Bockelman, D Bonacorsi, R Egeland, D Feichtinger, S Metson and J Rehn, Scaling CMS data transfer system for LHC start-up, 2008

QUATTOR: S Childs *et.al,* Devolved Management of Distributed Infrastructures With Quattor, 2008

SPLUNK: IT search engine, http://www.splunk.com

FSPROBE: P Kelemen, LCSC 2007 – Silent Corruptions, 2007