Joint DM-SM TEG workshop

[Day 2 - January 25th - Notes by *D. Bonacorsi* - v.1.0 - last update February 2nd]

Caches vs Archives (Daniele/Andrew)

Main Agenda: https://indico.cern.ch/conferenceDisplay.py?confId=165687
This talk: https://indico.cern.ch/getFile.py/access?sessionId=8&resId=0&materialId=0&confId=165687

Note: in the following "*{C,Q}/**Name***" refers to a *Comment* or *Question* by *Name*. In case the person who gave the {C/Q} is not clearly identifiable as from my notes, the field *Name* is set to *Unknown*.

Caches vs Archives

Daniele starts the talk (on behalf of Andrew Lahiff and himself) by reminding that this is a task in the original mandate that the SM-TEG received. From yesterday's discussion we could rephrase the topic as "Caches" vs "Archives" instead of "disk" vs "tape". On this topic, the LHC experiment workflows set the needs, but the implications on the sites and their set-up (Tier-1 sites here, since talking about tapes) are enormous. So, it's a joint responsibility, with boundaries to be discussed and defined (refer also to Dirk's main point as of yesterday's introduction on Data placement and Federations).

We have two classes of workflows at the Tier-1 sites on this topic:
- *READ* workflows. Need to keep defined samples on "performant" caches for reprocessing and redistribution. Need the ability to allow (limited or not) user analysis on sites having also custodial responsibilities without destructively impacting their "archival" system (i.e. make sure that users don't read plenty of files that must be recalled from tapes and hence put load on the tape system; users shouldn't write their data to tape, but that's easy to control e.g. by setting a file class appropriately).
- *WRITE* workflows. Need the ability to process samples without writing immediately to "archive" (e.g. (a) run reprocessing at one Tier-1 and archive at another, or (b) carry out validation before archiving)

The status of the {DM.SM}-TEG discussion on this follows. Surveys were don as a joint {DM,SM}-TEG effort on the 4 experiments, covering many topics, including this one. All experiments seem to be working fine with (or towards) splitting disk caches from tape archives:
- ALICE is split (they had a T1D0 only until some time ago, now split);
- ATLAS is split (they only explicit managed recalls from tape);
- LHCb is split (more info later);
- CMS not split yet, there's a work plan running though.

From the survey, HSM is felt as not to be "dropped", since it brings useful functionalities. One wants to process data in a HSM system, not have to manage a cache of data which is explicitly staged to another 'external' disk area. Also, functionalities connected to the concept of "recovering" files in a structured way. So, strategies towards managed split are either implemented already or going to be so soon.

- *Discussion*. C/**Brian**: what do you mean by "another 'external' disk area"?. C/**Philippe**: Experiments do not want to have to manage a cache of data that is explicitly staged to another disk area. E.g. when LHCb reprocess the RAW data, we do not want to have to export the RAW data from a custodial storage into a disk storage, we want to extract it and bring it to the WN directly from a HSM, so we want to BringOnline the data we need and then to pin it until we need it. C/**Brian**: so you want to "manage" it, actually, yourself? C/**Miron**: yes, you want to pin it, use it, unpin it.

Daniele continues, covering experiments one by one.

LHCb. Daniele quotes again specifically the LHCb case, using info from Philippe. They need 2 (and only 2) storage classes: T0D1 and T1D0. This is because no space (storage class) change is possible in some implementations of SRM. They made a full split between archive/disk for AODs (2 custodial replicas on T1D0, "n" replicas on T0D1, no T1D1 at all). Their RAW/ESD are write-once/read-few, don't foresee to have a copy of them on disk nor to replicate them on demand to T0D1 SEs prior to use, they would like to keep and rely on custodial storage with efficient disk caching. So, T1D0 has 2 functions for LHCb: archive of T0D1 and permanent storage of read-few data (e.g. RAW, RECO). For this the BringOnline functionality is mandatory. LHCb underlines that functionality for managing the disk cache should be supported by all implementations (in particular pinning, which is a *must*: sub-optimal usage of tape drives without it).

ALICE. They doesn't use SRM (hence no space tokens, etc). The ALICE SEs are in two categories:
- Tape (T1D0): presently used only for RAW data and accessible only to production users. Near future - store 'old' productions, which are not being analyzed
- Disk (D1T0): used by all (production, users) based on a storage auto-discovery model. More info here.

Regarding the disk-tape split. The tape volume is determined mostly by the amount of RAW to be written by the experiment (2 replicas of RAW - one at T0, one distributed at T1s) - thus it is deterministic and is specified in the ALICE requirements docs. The disk buffer for the tape system - this is largely at the discretion of the sites, depending on the operation policy - single disk buffer for all or separate per experiment (in the latter case, they determine together with the site experts what is the appropriate size of the buffer, usually it should be able to hold in one go (without need for re-staging) the entire Pb data volume from the current year stored at the site. For example at CERN this translates in a disk buffer of ~2PB). The disk SE gets the remaining ALICE allocation.

- *Discussion*. C/**Philippe**: ALICE wants to freeze 2 PB of disk at CERN, i.e. all the yearly data? This would not make sense, one just pre-stages at the beginning and then flashes the disk afterwards, cannot keep everything on disk all time. It needs some follow-up, and attention by the C-RSG. C/**Dirk**,**Daniele**,**Brian**: one year here means actually only the HI run, limited in time, to be able to have all the data quickly available for access and transfers all at once. Namely one-year worth of data here means one-month worth of data. C/**Philippe**: this needs to be understood and followed-up. E.g. LHCb asked T2D0 for RAW data at CERN, something like 700 TB, and we got oppositions. C/**Dirk**: point taken, this discussion should happen somewhere else, though. C/**Massimo**: this point has been scrutinized, so the C-RSG is involved. C/**Philippe**: it needs to be followed up. C/**Daniele**: the fact that one experiment needs to have everything recored in one year on disk gives us some input in a disk vs tape discussion, we have a clear direction of interest.. C/**Philippe**: of course this discourages us to put efforts on minimizing the resources that the experiments request C/**Daniele**: we might agree on this, but let's get back to the topic.

Daniele continues. To try to find key commonalities - but also not ignoring differences - among experiments, Andrew and Daniele decided to pick a multi-VO Tier-1 and just go through experiments supported on-site (e.g. all apart from ALICE), and we picked RAL. Differences at other T1s may apply, and site responsibles and experiment people were encouraged and welcome to comment at the workshop. So, the following focuses on the RAL-specific experiences.

ATLAS at RAL: The ATLAS set-up at RAL is based on separate space tokens for T0D1 and T1D0 (and no T1D1 at all), with different parts of the namespace associated with each. Having this distinction does not mean you have one token of each kind: you may have more physical resources and resources logically mapped to the same space token. In terms of jobs, you have the production jobs read from T1D0 or T0D1, and write to T0D1, while user analysis jobs can read from T0D1 only. Where does data on disk caches go? a) to other sites: vast majority of outbound transfers are from T0D1 (on occasions transfers come from T1D0), and b) to tape: data on T0D1 can be archived using FTS to copy from T0D1 to T1D0 (e.g. MC archived just before the next MC campaign is about to begin). The ATLAS operational experience with disk vs tape at RAL. Very-high speed FTS transfers from T0D1 to T1D0 was reported. (one can get a large tape migration backlog very quickly). Data can be written into the disk cache faster than data can be written to tape (small files contribute to this), so the disk cache can fill up, so you need need to have the FTS channel configured carefully.

- *Discussion*. Q/**Ian**: which is the % of small files that go to tapes? A/**Simone**: the output of the MC production is merged as much as possible. It depends on what you mean by “small”: we have files of the order of few hundreds MB (can be considered smallish). User files (even smaller) usually don't go to tapes (e.g. ntuples coming from skimming, slimming processes. Q/**Philippe**: you do not archive them? A/**Simone**: no. Q/**Daniele**: do you have a figure for the fraction of the total data considered “small” that go to tapes at T1 sites for ATLAS? A/**Simone**: it's not negligible, at some stage everything which is relevant for publications goes to tapes. Q/**Daniele**: so there is custodiality for not necessarily largish ATLAS files on T1 tapes only, correct? A/**Simone**: yes.

LHCb at RAL. We said more in general earlier. Specific details for RAL follow. The LHCb set-up at RAL is based on disk and tape separated. Where do jobs read from? Reconstruction and stripping jobs can read from T1D0. User analysis jobs read from T0D1. Where does data on disk caches go? DIRAC controls the copy (archive) of data written on T0D1 to the T1D0 service class (using FTS). In terms of LHCb operational experience with disk vs tape at RAL: no problems with this model, only minor issues very occasionally (e.g. FTS failures that need manual intervention).

CMS at RAL. The CMS set-up at RAL, currently, is that the system in production is almost entirely T1D0 still. So in principle (almost) everything goes to tapes ("almost" = not the tmp files in the so-called "merged" area, i.e. before they get merged). Where do jobs read from and where does data go? All jobs read from T1D0, write to T1D0 (apart from small temporary files, read above). All written files automatically go to tape, usually within 24 hours. Tests are being carried out with T0D1 and T1D0 service classes, and FTS to transfer files between them controlled by PhEDEx. Input data pre-staged by subscribing from T1D0 to T0D1. All jobs read from T0D1, write to T0D1. Output data archived by subscribing in PhEDEx from T0D1 to T1D0. As a consequence of this, at the moment there is no analysis at the T1 level for CMS - this may change soon once the split will be finalized. It must be noted that having a T1D0 has helped in numerous occasions at RAL. E.g. one issue which ATLAS has had a few times is the problem of some files on a T0D1 service class being unavailable while a disk-server is in intervention: CMS has largely avoided this because almost everything is on tape.

Some comments from the Castor team at RAL. Probably the best config indeed is to have 2 separate service classes (T0D1 and T1D0). They welcome FTS to be used to copy files between them. A note: in principle one could think of avoiding FTS transfers, e.g. in Castor at RAL you may be able to do DB manipulations to enable files to be initially T0D1 and then later migrate to tape (no FTS used), but please note this is *not* recommended and *not* supported at RAL. Possible discussion points raised: Any comments about using FTS here, or avoiding to use it (read above)? Status of the dCache feature request for allowing files to be written to disk and later migrate to tape if necessary?

- *Discussion*. C/**Ian**: if we are making a recommendations based on the concept that simplicity is better than complexity, the ability to manipulate the DB to move files back and forth between classes is a complication and it would be different among different technologies, while using FTS would be a little "brute-force" solution but it's very simple. So if there is no performance issue probably it's an acceptable solution. C/**Philippe**: we are talking about moves among classes which we are told e.g. by Castor that cannot be done, i.e. you need a way to delete the disk cache while keeping the migrated copies, you need to copy from T1D1 to T1D0, this is the main reason why we decided to have 2 copies, of course it has to be in different namespaces because you can't have 2 files in the same class with the same name.. but ok, we managed to do that, as Ian said this is maybe an overkill but it works, because things that should have been there were not implemented, now we have a way to do it and we are happy with that. The question is: how much does it cost to do FTS transfers internally, as a solution? C/**Miron**: this shows that the only tool that you have to schedule and manage internally resources is FTS, we have no other piece of software that allow us to say: we have to move this, put it into a queue, and manage it, and therefore FTS has been abused. The channel view is not necessarily the right one. We never created a scheduling system for managing the transfers, that can be installed locally. C/**Ian**: but for the actual transfer of data between classes, if you want to further separate the processing from the custodial storage, you can do it, while otherwise you can't. C/**Miron**: we just need a better data placement scheduler than FTS, and you are using FTS because that's the only thing you have. C/**Ian**: there is an FTS 3 in development since we know that FTS needs to evolve. C/**Michel**: but why you want two separate systems instead of just one? C/**Miron**: you need to basically schedule jobs. C/**Ian**: we could have used FDT, for example. C/**Daniele**: this is being tested and used in some places, and it's interesting, it has other implications though. C/**Ian**: right, so far we used FTS since it's the most simple for the use-case and promising. C/**Miron**: are there other queuing system that you can leverage to simplify and minimize the software that we develop and use? Maybe in a future FTS 6 you will implement a full scheduling system that exists in other places, with some special tricks because it's storage: that's my suggestion. C/**Michel**: understood the remark, but do not see why we should abandon FTS for something else. C/**Daniele**: I think Miron's point was about asking ourselves if FTS was not actually being sort of abused and if there was not instead other concepts or architecture that would fit the problem better. C/**Dirk**: we should not forget that the major point here is that FTS works for this, now. C/**Brian**: the layer, or the interaction point, between SM and DM on this specific topic is being moved up a bit to the DM (read: FTS) level, it seems. C/**Dirk**,**Daniele**: right, in the general data placement discussion, we are moving to consider FTS as a way to move data also among caches and archives, and not only among disks at different sites. C/**Simone**:

note that in ATLAS there is also data which are resident at T2s that needs to become custodial as well, and this is also done with FTS. The advantage of doing this with FTS together with T1-T1 internal data movement is that then you have a unique layer which is used to enforce the custiodiality, so ATLAS would strongly suggest _not_ to go to something separate for T1-internal data movement, because this would maybe improve performances a bit but it would complicate the overall system a lot, you will need to deal with two different systems, one for importing data into the T1 and one for the internal data movement. C/**Daniele**: rephrasing it, the interaction with the tape systems inside a T1 is not being considered to be different from any other source/destination in a transfer topology known by a LHC experiment data transfer system. C/**Simone**: exactly, it's the same workflow, so let's use the same tool and the same code. C/**Simone**: Another point is a clarification about the access to data on tapes by ATLAS. When we run the reconstruction, the data are read from the disk buffer in front of tapes, ATLAS prestages (SRM BringOnline) but then the data remains on cache, it's very similar to what Philippe said. For this, ATLAS requires or at least suggests that the disk buffer you use to put the data into the tape system is different from the one you use to transfer the data out of the site. C/**Daniele**: like the WANin and WANout pools at RAL. C/**Simone**: yes. Because you can have concurrent activities, and you need to support the stage-out for access together with the transfer inbound, for example. In some storage systems, this is enforced by definition, e.g. in dCache, in other system, e.g. Castor, you may need to set up different pools. This remains a requirements, i.e. the write not to be affected by the read in some ways. C/**Paul**: a comment on performances. If you look at FTS using 3rd party copy functions, maybe it's better to think of it as triggering the SE to do something, and might not be less performant than an internal copy: the change of a class reflects into a move in the namespace, so rather that saying the SE "copy that data over here" is more a "change of directory" so using FTS - provided you use 3rd party copy - does not give performance issues. C/**Miron**: the site still needs something, FTS or not, to give to the site-admins an extra handle to be able to fold in how they really want to configure the storage. Maybe, you also use cheap slow disk for example, and you need something to make sure the pool can handle the abstraction on the storage level, might not be a bad idea. C/**Paul**: it would be useful to have hints of when a data file is going to be read, so even if it is on tape or slow disks it can be fetched onto faster disks. C/**Wahid**: we need to understand the implications/recommendations on FTS and SRM, as a summary. C/**Dirk**: the important thing is that we understood that we can use FTS to do the move/copy, and what FTS needs to operate below is a separate question. C/**Ian**: coming back to the original discussion, essentially FTS here is serving the capability of a pre-stage command. C/**Simone**,**Philippe**: no, the pre-stage is from T1D0 to T0D1 which is rare (...) in ATLAS the data are read directly from the disk buffer in front of tapes, not moved to T0D1 before you read them. Q/Jeff: but you are changing the classes because you want to change it from archive to something like "analysis"? A/Philippe: no, you want some data that is useful for analysis which is on T0D1 and you want it to be archived because it has been used for publication and you want to keep an archive, otherwise we can just keep it on disk. C/**Daniele**: but this is actually symmetrical, no? C/**Philippe**: that's more rare.. C/**Ian**: FTS is used for pinning, that's the other direction. C/**Philippe**: no, we are confusing the whole story now. Copying from T1D0 to T0D1 is a recovery operation in my mind, the fact that CMS is using this paradigm for something else.. C/**Ian**: but you transfer it to T0D1 from some place, you pin the data on disk. C/**Philippe**: yes, T0D1 is pinned forever. C/**Ian**: right, and whether you staged this from a tape archive or somewhere else, you said "this data needs to be on disk", so you are using the existence of a T0D1 cache pool for this. If you are using FTS to populate it from the outside, and you are using FTS to transfer it back to T1D0, the fact that they are the same storage is unnecessary. C/**Philippe**: it could be EOS and Castor. C/**Simone**: it's like this for ATLAS. C/**Ian**: so, the fact these storage classes T0D1 and T1D0 simply define a functionality we can simply call it cache and archive. C/**Miron**: again, what FTS does is resource provisioning, it whole idea was not to overload the channels, I open 5 channels into my storage and there will always be 5 going. This is about resource provisioning, not content delivery, and it should evolve towards a scheduler: fundamentally I do not see differences between scheduling other jobs and data placement jobs, they are jobs like all other jobs (whom do you you belong to, how many resources you have, how many did you run, what's your fairshare, etc) and it will evolve to this.