- We assert that we have a working system, do not need a new system, but the current system (cost and complexity) exceeds what is needed. Would the experiment agree with this statement?

The system could be simpler, but the main problem is the duplication of effort between generic middleware providers and each of the experiments. We suffer also from the fact that standards are not properly implemented.

- What elements are being used, and how? Which have been added and why?

We use FTS, the Storage Elements (SRM), the LFC, lcg_utils and gfal. A DIRAC Replica Manager has been added to hide the Storage Elements incompatibilities, to implement the retries, quotas and the LHCb policies. Missing functionality of the middleware was implemented in DIRAC agents (for example du catalog functionality for implementing quotas). Any change of functionality (e.g. adding registration capabilities in FTS, gfal2) requires modifications in the upper-middleware (DIRAC) and should be properly discussed before being implemented.

- Which components failed to deliver (larger) functionality needed? Does that mean we can deprecate and obsolete these components, or was it not truly needed?

LHCb does not use BDII in DM and implemented its own Information Service for static information. SRM 2.2 extensions are not implemented properly by all implementations (e.g. ACLs, pinning). The main usage of SRM done by LHCb is as an abstraction interface to storage, but most features are used (not dynamic space reservation as it is not really implemented). Our main reproach to SRM implementations is that they were considered by developers (Castor and dCache) as add-on that should adapt to the existing storage infrastructure, but not as requirements. No changes were made essentially in the core of the storage systems.

- If the middleware dropped complexity and removed functionality, does the experiment have the resources to adapt to the change?

Basic functionality should not be dropped. Introducing backward incompatibilities is always bad because the experiment manpower in scarce. Of course we would have to find manpower to implement missing or declared "obsolete" features. Gfal2 is a perfect example of undiscussed change. Functionality is incomplete and not discussed with users.

- What is the experiment's interest in revising data placement models? What kinds of revisions have gone through? What is the experiment's interest in data federations? What sort of assumptions does the experiment need to make for data federations work?

We want to use a dataset popularity system for sizing the number of replicas of datasets but still schedule the jobs where the data is. Replicating data on demand should be limited, otherwise the system may not scale (network and Storage Elements). The popularity is seen primarily as a tool for the Data Manager to manage disk datasets.

- Could you work directly with clustered file systems at smaller sites?

Not relevant for LHCb as we use Tier2s mainly for simulation. Data processing use cases use the copy-to-WN paradigm, therefore do not require a local SE.

- Could you work directly with cloud file systems (i.e. GET/PUT)? Assume "cloud file systems" implies REST-like APIs, transfer via HTTP, no random access within files, no third-party transfer, and possibly no file/directory structure. See Amazon S3 for inspiration.

We use hierarchical name space, HTTP can be used for root I/O access, but we would still need third party transfers for file replication.

- How thoroughly does the experiment use space management today?

We use space tokens for the different Service Classes, but we had to implement tools for usage and quotas as this functionality is missing in the basic/common components. We had to move from functional space management (one SRM space for each type of data) to this mode due to severe limitations in data access performance in the former model (physical identification of spaces to disk pools for example)

- Does your framework support HTTP-based access?

We do not currently use it but could enable it in root I/O. Performance is most likely the issue. Everything else (data transfers) would need development of DIRAC plugins.

- Is the archive/disk split an agreed-upon strategy in your experiment? Can HSM be dropped?

The point is not clear. We made a full split between archive/disk for AODs (2 custodial replicas on T1D0, n replicas on T0D1). We don t have any T1D1 storage class. However RAW data as well as ESDs are write once / read few types of files. We don t foresee to have a copy of them on disk nor to replicate them on demand to T0D1 SEs prior to use. Therefore we would like to keep custodial storage with fast and efficient disk caching. Functionality for managing the disk cache should be fully supported by all implementations (in particular pinning).

- What role do you see for Data Federation versus Data Push? Is this useful at smaller sites?

Not relevant for LHCb as we use Tier2s mainly for simulation.

- For smaller sites, would caching work for your experiment?

Not relevant for LHCb as we use Tier2s mainly for simulation. It would however be an option replacing formal data placement on analysis centres (mainly Tier1s). It is not clear whether caching would be better than formal placement with a dedicated system for increasing/decreasing he number of replicas based on data popularity.

- Security / VOMS - what are experiments expectations / needs.

Current model would be ok if it had been adopted by all Storage Element providers. For the moment we get the permissions from the LFC and have no real way to close the backdoors, which could lead to SE/catalog inconsistencies.

- Where they see their namespace management "philosophy" evolving in the future.

Users only see the logical namespace. The physical namespace conventions are mainly coming from sites requirements for defining tapesets (for T1D0). A similar structure was then applied for the other datasets quite naturally. But we could replace the PFNs with GUID and our system would work as well. Human readable names and structured set of directories are however preferred. Each file has a single LFN. Jobs use LFNs to refer to input files. However if ancestor files need to be accessed from the application, the GUID is used (stored in the daughter file) and the dereferencing is done using a local XML catalog built when the job starts. If service attributes were attached to a directory, our new usage of SRM spaces would make no difference, since we renounced changing space for a file. SURLs are created by DIRAC as follows : srm:// , and are part of our configuration data. The SURL registered in the LFC is not used.

- What volume of data do experiments plan to move / store (and what do they currently move /store).

???. About 150 to 200 MB/s aggregated on average

- What kind of file access performance are they expecting (any changes expected?) - what WAN data transfer rates?

We do not use WAN for file access, only for replication between sites (150 MB/s on average). But we do need to have enough disk spindles to support the number of concurrently running analysis jobs. This is the main limitation today.

- Can we get rid of SRM in front of our storage services? (question also for Storage group)

It is the only (common) interface to Storage Elements and currently the only way to do pre-staging and pinning on dCache.

- Why do we need gridftp? Why can't we use http like the rest of the world. (NB: "because it can't do xx" is not a good answer - we are good at developing something from scratch and bad at adding functionality to open source sw).

We do not see many problems with gridftp except for downloading data to WNs where multiple streams cannot be used (requires incoming connection from the SE). However if FTS and/or lcg-cp would make use of another transfer layer, it would be transparent for us (provided the performance is there). We also see very little success in having WLCG proposed modifications adopted by opensource software providers.

- Can we agree layered data management and storage architecture? This will help factorise various issues and help piecemeal replacement of layers. To be defined together with storage group

As agreed long ago with WLCG, we used the proposed layers architecture: FTS on top of SRM for transfers, lcg_utils and gfal as API to the SRM library (using python binding). If these interface do not change or are negotiated long in advance and provided the same (or better) functionality, there is no problem in changing the implementation of the layer. The first thing to do would be that when an interface is defined, it gets implemented by all providers (cf SRM).

- Is HSM model needed? (etc see Dirk's GDB slides): NO?

We do believe that tapes are useful as we will never get enough disk capacity, not to mention that cheap disks as we buy now will probably soon not be manufactured. Pre-staging model (poor man s HSM) is probably enough. All T1D0 datasets are accessed by production jobs downloading to the WN. Therefore we see no reason to have to replicate them to a T0D1 storage for mostly a single use. This is an additional burden that we don t want. However in order to efficiently download files from the HSM to the WN, we need both an efficient network and enough spindles on the disk cache.

- How will we manage a global namespace - whose problem is it? The Experiments? Is there a continued need for LFC?

Even with a global namespace, we see little advantage in not scheduling sites for jobs, therefore we probably need always a replica catalog for job brokering and (very important) data space management (we don t see how to control an unmanaged system). We use only a subset of LFC: namespace, permissions, status and list of sites having a replica of a file. We will test the DIRAC File Catalogue as well. DIRAC is building the SURL from the static storage information and the LFN. The catalog is before all a replica catalog but very importantly this is the place where permissions are checked. Both LFNs and GUID are used by the application framework. Only LFNs are used in the DMS.

- Access control - what is really needed? The simpler the better!

We rely on VOMS and LFC for the authorization. Again, it's not enough to define or adopt a standard, it must be implemented by all software providers. We do not see a need for changing the current model.

- As a general comment, we would like a better clarification/division of responsibilities between experiments and infrastructure(s). I.e. if the experiments want to manage most of the complexity then we should simplify the services that the infrastructure should provide.
    - Posed as a question: Where should the complexity and intelligence lie - the experiments or the infrastructure? How do you view the balance now, and how would you like to see this change (if at all)?

It is too late to ask this question. The experiments have already implemented what was missing or not working in the generic solutions.

-- PhilippeCharpentier - 06-Dec-2011

This topic: LCG > WLCGTEGDataManagement_LHCb
Topic revision: r4 - 2012-01-24 - WahidBhimji