# Data Management Futures

## OSG Data Management

Traditional data management on grids like the OSG involves policy-based distributing files or datasets to multiple sites, registering the location of these datasets, then sending jobs to the data’s location.

CMS and ATLAS currently implement the “full chain” of data management, but others have found it outside their means. Both are currently exploring a more dynamic cache-based policy as a means to more efficiently distribute data according to demand, decrease failure rates due to missing files (a data loss rate of 0.01% can be a huge operational load), and decrease the operational costs (especially at smaller sites).

Other data management approaches include:

* Dzero had an pre-existing cache-based data management system which was adopted to use SRMv2.
* LIGO uses a combination of Globus RLS, home-grown catalogs, and Condor to transfer files to sites. However, due to their requirements of opportunistic storage, POSIX access and consistency issues, they have only been able to run at 1-2 sites even if they can transfer to more sites than that.
* Engage (as a mixture of projects) has sometimes used a semi-automated approach of transferring files on a site-by-site basis.
* HCC and SBGrid utilize a centralized model, where all inputs are staged in from a large site (possibly cached on worker node disk between jobs) and staged back out to the large site.

## State of the Art

The current OSG infrastructure is competent in the following things:

1. Providing scalable storage endpoints.
2. Relatively consistent implementation of a subset of SRM.
3. Transferring “large” files between two sites upon request via standard protocols.
4. LAN-friendly protocols for the local site.
5. Providing a storage namespace at each endpoint.
6. Interoperability with the WLCG.
7. Storage service discovery.

Large VOs who own sites and want to use them to store data can efficiently move data .

The following is missing in our current data management framework (meaning it is often re-implemented by each VO):

1. Uniform storage management capabilities for sites.
2. VO tools for storage management.
3. POSIX (or POSIX-like) local access.
4. Site consistency tools.
5. High-level transfer management tools.
6. Replica management tools.
7. Catalog tools.

The goal of the OSG Public Storage Service was to provide (1,2). We see (3) requested often by new and non-LHC VOs. Item (4) is handled on a site-by-site basis. Items (5,6) are provided by EGI but OSG ships the clients. Item (7) is implemented by each large VO. The first three are considered “low-level” items while the latter three are “high-level”.

## New Directions for Data Management

OSG has a considerable amount of work to do if it is to implement the traditional grid data management framework for its non-LHC VOs. Assuming enough resources to accomplish this (and assuming it’s within our scope), we must consider the fact that LHC VOs are beginning to adopt new models of data management. OSG must adopt with the USLHC system in order to provide support.

The new LHC data management models are a hybrid of the explicit policy-based data placement and cache-based models. The T0 and T1 LHC sites will continue to have strict policies, especially for archival storage. The T2 sites will be able to cache files based on realtime demand in additional to the current policies. The T3 sites will likely turn to completely cache-based sites.

ATLAS has been able to convert its experiment-specific job and data management systems to identify when jobs are waiting in queue, trigger data movement, and then rebroker jobs as possible to the new site.

ATLAS, ALICE, and CMS are also investigating experiment-independent cache-based systems utilizing the Scalla software suite from SLAC (often referred to by ‘xrootd’, and utilizing the xrootd protocol). These are based on placing large disk caches throughout the grid and linking them using a centralized redirector. Users outside any grid site can contact a central endpoint and be redirected to the closest file replica. From within a site, the user job will attempt to open the file through the default mechanism (POSIX, dCap, or xrootd). If the file is not found, the site can be configured to instruct the clients to stream the file directly from the remote site, or pull the file into its cache.

We foresee the WLCG VOs building a tree of Xrootd servers. The leaves of the tree are actual data servers, while the others are redirection nodes. For example, the root of the tree would be a global redirector, followed by a layer of per-grid regional redirectors, then site-local redirectors, then finally the data servers. In order to prevent clients from querying the entire grid, they will default to the deepest redirector in the tree. The redirector will broadcast the data request downward, then only hand off to the higher layer in the tree if the request cannot be satisfied.

We believe such a system can be generalized to benefit other users of the grid. It eliminates the need for VOs to build data-transfer systems or maintain location catalogs. It provides a simpler, homogeneous access layer for applications, as the file is referred to by the same logical name throughout the grid. Xrootd has the ability to keep several distinct caches (possibly one per VO), although it lacks in terms of dynamic space management as was dreamed about in SRM.

Xrootd does not replace the need for a large VO to have data cataloging and archival. Archival sites tend to be more reliable for their associated VO and much smaller in number. Within the last 2 years, iRODS has become a popular data cataloging tool, and we have been experimenting with it to see if it can fill this role for OSG VOs.

Short term plans

* **Assist Continued LHC adoption**: The LHC will continue Xrootd demonstrator projects. It appears they will be adopted by LHC T3s and officially by the experiments themselves.
  + OSG will need to assist the LHC with WLCG interoperability. Endpoints will need to be monitored with RSV and registered in OIM. Like the current relationship between OIM and OSG CEs, this is not needed for operation of the endpoints themselves, but for keeping it as a part of the production infrastructure.
* **OSG as a software packager**: As the technology has been demonstrated, we believe the OSG will take an increasing role in packaging and deploying the technology in its grid.
* **OSG as a support mechanism**: OSG needs to be well-positioned to help sites transition to this new architecture.  OSG will need to play a support role for system administrators in deploying Xrootd and VOs who want to adopt it.

### Long term plans

Once packaged by the OSG, we will start to focus on deploying Xrootd out to all sites.  If available opportunistically at many large sites, we believe it can start to be adopted by non-LHC VOs. The mechanisms learned by LHC will ideally “trickle down” to the smaller VOs.

With Xrootd and caching, we significantly decrease the layers that currently must be written by the VO (replica management, transfer management) and improve the overall efficiency/reliability (missing files are silently re-transferred and overall processing efficiency could be improved if data can be moved to match available job slots). With the possible addition of iRODS for catalog management, OSG could offer a complete data management stack.

### Technical Issues and Ideas

We believe the following technical issues need to be addressed or solved in the final system:

* **Limit worst-case behavior:** If a VO’s active working set is larger than the site’s cache size, we can end up with worse scalability than we currently have.
  1. **Opportunistic storage VOs:** The current worst case for a VO using opportunistic storage is inefficient operations. Jobs may sit in queue because the data isn’t distributed at enough sites. The worst case in the proposed caching scheme is that every job stages in all its data.
  2. **Centralized data distribution VOs**: Currently, centralized data distribution VOs transfer all files over the network each time. The worst case for caching is the same network pattern, but bottlenecking all the transfers through one cache node.

Both worst case scenarios occur when the working set size of the cache is greater than the cache size. There is no clear mechanism to alert the VOs when thrashing occurs, but the caches themselves should be able to determine thrashing behavior. We propose to monitor the cache eviction rate and to notify sysadmins when the daily average rate goes above a threshold. This later could be modified so the local Xrootd redirector would know when the eviction rate is too high and automatically turn off caching (using remote streaming instead).

* **Prefetch policies**: The currently-implemented prefetch policy is simple – when the file is opened and a cache miss occurs, the local redirector can chose to redirect client to the global redirector (so the client will stream the file for the rest of the session) or cache the entire file. Clients can request the local redirector prepare files; this asks the redirector to make the “stream or cache” decision for a set of files at once. CMSSW currently “prepares” all files it will read when the process starts.  
  CERN is exploring caching subsets of the file based on demand, under the idea that clients will often repeat data access patterns. It is not immediately clear removing the prefetch of the entire file will be beneficial. We will be following the progress in this area.
* **Cache eviction policies**: Cache evictions are based on an LRU policy for each server. It is technically possible to implement a new eviction policy, but this is not a modular part of Xrootd, and we are currently of the opinion it is best left for the Xrootd developers. We believe that implementing bottlenecks to limit the worst-case behavior is more important than picking a cache eviction policy designed for a particular VO or usage pattern.
* **Client I/O patterns**: LHC clients sometimes only read out a small subset of the file (there are jobs which correspond to reading 1%, 10%, or 100% of the file). Much of the research into I/O in the past 12 months has focused on improving the read patterns. Currently, CMS and ATLAS jobs can declare beforehand the data they will be reading for many job types, and work is being done to make the patterns better-behaved and more predictable.

Some of the work can be done in the server to do caching only the sub-selections or caching small chunks at a time. Neither option has shown to be a general improvement when all use cases are considered. We believe it will be difficult for the server to do better predictions of selections than the client due to the ROOT file format and the amount of logic built into the client.

* **Cache-aware job scheduling**: The data placement system and the job scheduling system are roughly integrated already. Currently, when the jobs are created, they are limited to the list of sites that already have the necessary data. The list of OR’d sites is passed to the job scheduling system, which then selects the optimal ones. There are many possible schemes to do a tighter integration of the two systems. For example, the job scheduling system can prefer sites where the data is already pre-placed, but choose to schedule jobs to otherwise idle resources that can quickly cache the data.  
  The job scheduler is one area where limiting the worst case scenario is important – if jobs were distributed through the grid completely at random, caches would be almost certain to thrash.