StashCache – Distributed Data Access for the OSG

The goal of this project is to support all OSG users who have common input data files, with an initial goal of handling up to 1 TB datasets. Users will upload copies of their data to predefined sources and will set up jobs to fetch data without having to understand implementation details. Internally, this will rely on a hierarchy of caches - based on XRootD[[1]](#footnote-1) – to serve files.

The StashCache project will build upon the existing OSG Campus Infrastructures implementation.

# Problem

The two large LHC VOs, ATLAS and CMS, own storage at many OSG sites and use them as *storage elements*, or remotely accessible file systems. The storage element concept does not work for opportunistic VOs – without a mechanism for automated eviction, files stay in the storage element forever. Without access to site storage elements, opportunistic VOs cannot efficiently deliver files to individual jobs. The CMS and ATLAS systems for moving data between storage elements are robust and efficient, but have proven impossible for any other VO to operate.

Users not affiliated with ATLAS and CMS have few remaining choices for distributing data files, especially as dataset size increases. Currently, most of these users rely on HTCondor file transfer from submit to execute host, using on-site HTTP caches, or OASIS. All three mechanisms effectively limit the size of data files to less than 1 GB. In 2014, over 30% of OSG cycles went to non-HEP users, and anecdotally at least OSG knows that some of those users have common input datasets that exceed the best-practice limits for available methods.

The general problem is quite complex; we choose to focus on “shared input data”. We assume the user is running at least 1,000 jobs (lasting 1 hour each) across a shared input dataset of at least 1 GB.

# Architecture

Macintosh HD:Users:bbockelm:Documents:AAA:StashCacheOverview.pdf

The basic architecture is illustrated above. A **Source** is a location where users upload their shared input data files, with the intent that users upload any unique file just once for a given workflow. Each VO source connects to a central OSG redirector; the VO source is managed by the VO, while the OSG redirector is managed by OSG Operations.

As part of the environment setup, the pilot will determine the location of the nearest cache. Payload jobs will attempt to download the shared input files from the local cache; if the file is not in the cache, the cache will contact the central redirector to find the file. The redirector, in turn, will contact the appropriate VO Source for a given file. Based on the source response, it will redirect the cache to the appropriate source server. The requested data is transferred from source server to cache, then from cache to the client; the cache will potentially download the complete file and save it to disk for future requests.

For the pilot project, we will collaborate with the OSG Connect to verify the cache size is larger than the size of the possible input files (in order to avoid cache eviction). Past the pilot project, we will start with a least-recently-used eviction policy and work with the AAA and HCC teams to develop software for managing cache thrashing. Each VO will manage their VO Source contents; we will provide no specific management tools.

We assume files in this system are immutable and world readable.

For the pilot project, we will have only one source: OSG Connect.

# User Interface

**Adding files to StashCache**: To add a file to the StashCache system, a user will copy it to the StashCache source for their VO. Each VO will manage their source independently; the OSG-Connect integration will be a mounted filesystem on the login host. An OSG-Connect user will execute “cp” to copy the file from their home directory into their user directory in Stash. As this is a normal POSIX filesystem, the user can utilize the familiar set of POSIX filesystem utilities for management. VOs may layer on other management mechanisms, such as Globus Online or a HTTP interface, to their source.

For accessing files (particularly from a user’s OSG jobs), StashCache will provide three mechanisms:

* **“cp**”-like: We will provide a tool, *stashcp*, which will emulate the “cp” interface to copy a file from StashCache to the local disk.
* **HTCondor File Transfer**: HTCondor copies the file to the working directory if an appropriate URL is specified in the job submit file.
* **POSIX**: The job can access StashCache files by doing POSIX IO (open, close, read, write, etc) in their executable against a specific directory.

One explicit goal of the StashCache project is that the user never sees the implementation technology (Xrootd) – only the above pre-defined interfaces. They *may* need to know their VO’s namespace. That is, if they refer to the file as /mnt/bigfiles/joe/foo on their login host, they will need to know StashCache refers to the same file as /stash/osg-xd/joe/foo.

# Component Architecture

**Source server**: Each participating VO or team will have a single “Xrootd source” server that they operate. The VO is responsible for this server and managing the namespace (i.e., setting up user directories). The OSG will provide packaging and default configuration files for this service. The default VO Source implementation consists of running and configuring the xrootd and cmsd daemon from the xrootd-server RPM package.

**OSG Redirector**: An Xrootd redirector service will be run at the GOC. For the first year, this will be a “hosted service” in the manner of the GlideinWMS factory; it will be run and managed by experts.

**Cache servers**: Multiple caches will be deployed at sites distributed strategically throughout the OSG; we will target servers with 10Gbps interfaces and >10TB of storage. The cache server software will be based upon a custom deployment of Xrootd using the caching proxy plug-in; OSG Software will provide the customization. Operations will be done jointly by the hosting site and the OSG.

**Runtime software**: No software or services is required at the site. The client environment will be setup by the pilot; the client software (the xrootd client, plugins, and stashcp wrapper) will available through OASIS. Post-pilot-project, we will provide a mechanism for accessing StashCache on hosts without OASIS access.

1. http://xrootd.org/ [↑](#footnote-ref-1)