Engineering the CernVM-Filesystem as a High Bandwidth Distributed Filesystem for Auxiliary Physics Data

2015 J. Phys.: Conf. Ser. 664 042012

(http://iopscience.iop.org/1742-6596/664/4/042012)

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

FERMILAB-CONF-15-211-CD
ACCEPTED

21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015) IOP Publishing
Journal of Physics: Conference Series **664** (2015) 042012 doi:10.1088/1742-6596/664/4/042012

Engineering the CernVM-Filesystem as a High Bandwidth Distributed Filesystem for Auxiliary Physics Data

D Dykstra[1], B Bockelman[2], J Blomer[3], K Herner[1], T Levshina[1] and M Slyz[1]
[1] Scientific Computing Division, Fermilab, Batavia, IL 60510, USA
[2] University of Nebraska-Lincoln, Lincoln, NE 68588, USA
[3] PH-SFT Department, CERN, Geneva, Switzerland

E-mail: dwd@fnal.gov, bbockelm@cse.unl.edu, jblomer@cern.ch, kherner@fnal.gov, tlevshin@fnal.gov or mslyz@fnal.gov

Abstract. A common use pattern in the computing models of particle physics experiments is running many distributed applications that read from a shared set of data files. We refer to this data is *auxiliary data*, to distinguish it from (a) event data from the detector (which tends to be different for every job), and (b) conditions data about the detector (which tends to be the same for each job in a batch of jobs). Relatively speaking, conditions data also tends to be relatively small per job where both event data and auxiliary data are larger per job. Unlike event data, auxiliary data comes from a limited working set of shared files. Since there is spatial locality of the auxiliary data access, the use case appears to be identical to that of the CernVM-Filesystem (CVMFS). However, we show that distributing auxiliary data through CVMFS causes the existing CVMFS infrastructure to perform poorly. We utilize a CVMFS client feature called "alien cache" to cache data on existing local high-bandwidth data servers that were engineered for storing event data. This cache is shared between the worker nodes at a site and replaces caching CVMFS files on both the worker node local disks and on the site's local squids. We have tested this alien cache with the dCache NFSv4.1 interface, Lustre, and the Hadoop Distributed File System (HDFS) FUSE interface, and measured performance. In addition, we use high-bandwidth data servers at central sites to perform the CVMFS Stratum 1 function instead of the low-bandwidth web servers deployed for the CVMFS software distribution function. We have tested this using the dCache HTTP interface. As a result, we have a design for an end-to-end high-bandwidth distributed caching read-only filesystem, using existing client software already widely deployed to grid worker nodes and existing file servers already widely installed at grid sites. Files are published in a central place and are soon available on demand throughout the grid and cached locally on the site with a convenient POSIX interface. This paper discusses the details of the architecture and reports performance measurements.

1. Introduction

Fermilab has several physics experiments including NOvA, MicroBooNE, and the Dark Energy Survey that have distributed applications that need to read from a shared set of data files. We call this type of data *auxiliary data*. We consider this data distinct from (a) event data (which tends to be different for every job), and (b) conditions data (which tends to be the same for each job in a batch of jobs). Conditions data is on the order of hundreds of megabytes per job where both event data and auxiliary data are larger per job (multiple gigabytes). Unlike event data, auxiliary data comes from a

Operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy.

limited working set (tens to hundreds of gigabytes) of shared files. Since there is some spatial and temporal locality of the auxiliary data access, it appeared distributing auxiliary data would need infrastructure similar to the CernVM-Filesystem [1] (CVMFS) infrastructure; that infrastructure was built for distributing software to the grid. In the software distribution use case, because grid jobs tend to be started in large batches running the same code, the existing CVMFS infrastructure has a very high cache hit ratio; further, requests are very often satisfied from the client's local disk cache. Accordingly, the bandwidth requirements on the per-site HTTP proxy caches is relatively low. Most sites use a small number of 1 Gbps-connected hosts. In addition, the high cache hit ratio on the proxy caches enables them to make good use of filesystem in-memory buffers, resulting in low disk bandwidth requirements. As a result those proxy caches have been engineered with relatively low bandwidth and slow disks. In our tests, they are easily overwhelmed by auxiliary data which has a lower cache hit ratio and larger working set size.

This paper describes a new approach to the problem that still uses CVMFS software but takes advantage of a CVMFS client feature called "*alien cache*" to cache data on a site's existing high-bandwidth data servers. This allows us to utilize a different data distribution infrastructure. We have tested this use of alien cache with the dCache [2] NFSv4.1 interface, Lustre [3], and the Hadoop Distributed Filesystem (HDFS) FUSE interface [4], and found that they all perform well. In addition, we use high-bandwidth data servers at central sites to perform the CVMFS Stratum 1 function instead of the low-bandwidth web servers deployed for the CVMFS software distribution function. We have tested this Stratum 1 function using the dCache HTTP interface.

Section 2 of this paper discusses the details of the problem, Section 3 describes the architecture of our proposed solution, Section 4 reports performance measurements, and Section 5 provides details of the implementation.

2. The auxiliary data problem

For a prototypical auxiliary data application, we use a physics calculation application called GENIE. GENIE is currently in use by multiple Fermilab neutrino experiments and serves as our benchmark. This application uses a configurable number of jobs, each of which randomly select a subset of a larger dataset to read. While configurations vary, we used as a benchmark a 15GB dataset and run 128 jobs. Each job reads about 2GB of files (with about 140MB in each file). Within the application, these files are called *flux files*. Each job downloads its subset of flux files at the beginning of its run and re-reads them a few times during the course of an eight hour run. Our 15GB dataset is real but relatively small; other datasets are up to 250GB.

The performance of the benchmark was first measured on an ordinary CVMFS installation; the measurement was done after all client caches were invalidated. The site had four HTTP proxies, each with two bonded 1 Gbps network devices (2 Gbps total) and each with two application processes with separate disk caches[1]. Figure 1 is a plot of the amount of time it took to transfer the 140MB files according to the proxy logs. Note that this was not the total time for a job to read flux files; each job read enough of the files to get up to about 2GB. During the peak load time, it was taking nearly 1,000 seconds, over 15 minutes, to transfer one of those files. Many different files were being loaded at once by different worker nodes and the working set size was larger than the system memory, so the kernel in-memory filesystem page cache was not used to any advantage. All the data had to transfer from the machines' slow disks. The 128 jobs ended up on many different worker nodes so there was minimal sharing of the worker nodes' on-disk CVMFS client cache. Even if all 128 nodes were reading a different 140MB file simultaneously, at a rate of 1,000 seconds per file, the aggregate

[1] The HTTP proxy implementation used throughout this paper was Squid 2.7 from http://www.squid-cache.org.

bandwidth over all four machines would be 18MB/s. Other benchmarks of reading one reasonably sized file repeatedly from the same machine (the best case) showed them each capable of 200MB/s (for a total of 800MB/s). Hence, the application use pattern – highly concurrent access spread over a small number of disks – causes the performance degradation.

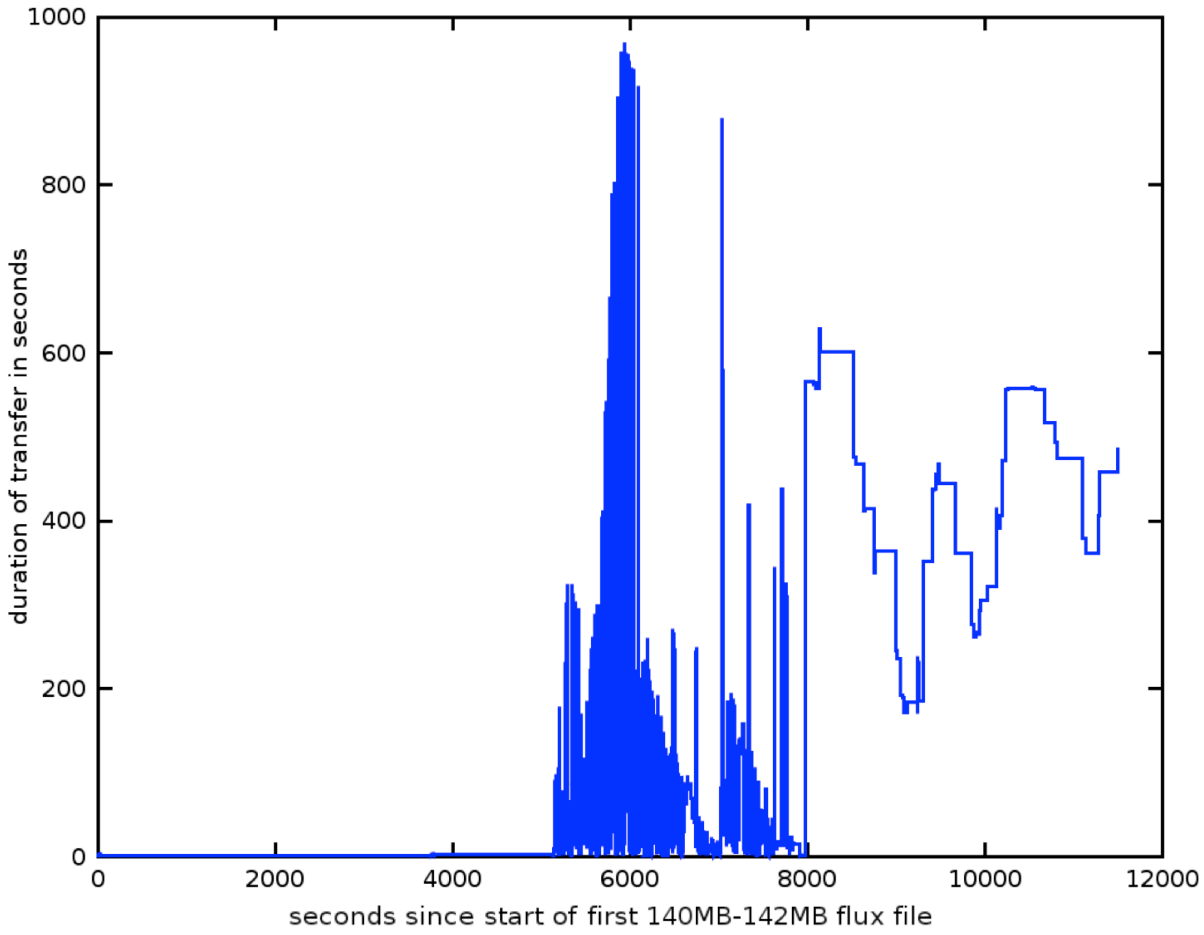

Figure 1. Plot of time to transfer flux files versus transfer start time

Another concern with using ordinary CVMFS for this application is the amount of available worker node disk cache. A small dataset is 15GB, but a large dataset can be much bigger (250GB). As the recommended CVMFS worker node cache size is only 20GB for all users, we run a significant risk of cache thrashing. Another solution for auxiliary data is needed.

3. Using CVMFS with high-bandwidth Storage Elements

Although the ordinary CVMFS infrastructure is engineered for distributing software and so does not have high bandwidth or fast disks, there is another piece of infrastructure common at large computational sites that does have high bandwidth and fast disks: the *storage element*. The storage element, often exposed as a POSIX-like filesystem, is engineered to store and transfer multiple petabytes of event data – significantly more scalable than the HTTP proxies. A CVMFS feature, *alien cache*, replaces the local disk cache with a shared POSIX-like filesystem. We replace the local disk cache with site storage elements. In addition:

1) To not interfere with the traditional CVMFS use case of software distribution - but still make it easy for client configurations - we configure a separate domain for data repositories. For

example, we would set up a domain called *osgdata.org*. Using a new domain, sites can setup multiple repositories with a single configuration change (CVMFS repository domains do not need to correspond to DNS domains).

2) We require sites to mount their storage element on their worker nodes and make an area writable by the *cvmfs* user id; sites then configure the *osgdata.org* domain to use this space.
3) Worker nodes disable HTTP proxies for the CVMFS data domain; the shared storage will be used as the only site cache.
4) We also made a new CVMFS Stratum 1 based on high-bandwidth storage elements. This is essential for transferring large datasets.

Figure 2 compares the design of traditional CVMFS to our design for auxiliary data.

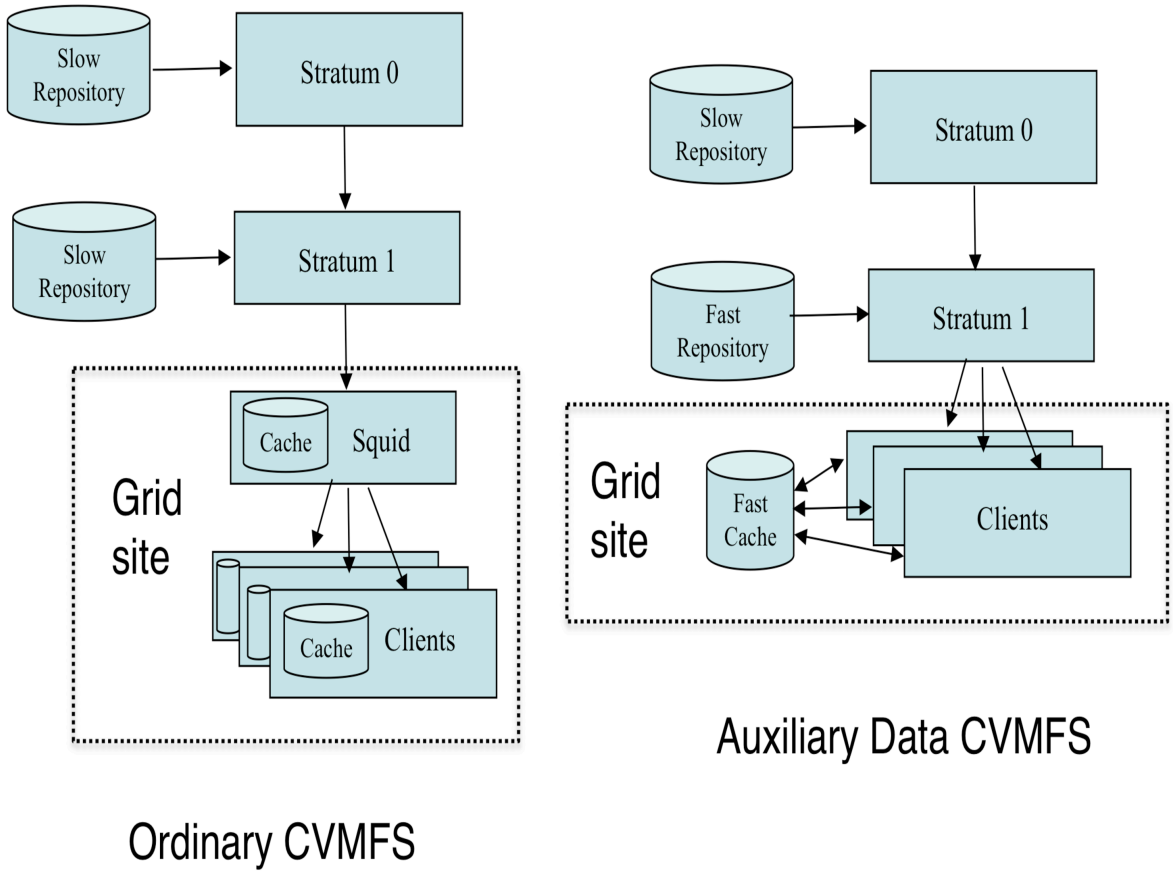

Figure 2. Ordinary CVMFS design versus the auxiliary data CVMFS design.

As Figure 2 shows, the slow disks on Stratum 1s, site squids, and clients in an ordinary CVMFS infrastructure are replaced with fast file servers, one for the Stratum 1 and one at each grid site.

As there is no proxy server in the auxiliary data design, it is possible for more than one client to request the same file at the same time before it is in the alien cache. In that case, each client independently downloads the file from the Stratum 1, writing to temporary file names. The first client finished will rename it to the final name in the cache; the other copies get discarded. This may cause wasted bandwidth, but we assume concurrent downloads are relatively rare and, as the maximum cache file size is 64MB (as explained in Section 5 below), relatively fast. Any clients later requesting the same file will read from the alien cache copy.

This design is only intended for relatively large data files as storage elements are generally engineered for large files and there is extra network traffic for clients for every alien cache access (compared to client local disks in the ordinary CVMFS use case). As CVMFS benefits from the kernel page cache, if multiple jobs on a single worker node access the same file, it may be served from the page cache (avoiding network traffic). However, as pages are quickly flushed from the kernel cache, we would still expect significant network traffic for ordinary CVMFS usage patterns.

4. Performance

We have tested our auxiliary data CVMFS design using three types of storage elements: Lustre, HDFS, and the NFS v4.1 interface to dCache. Figure 3 is a comparison of timings for three different configurations of the benchmark described in Section 2.

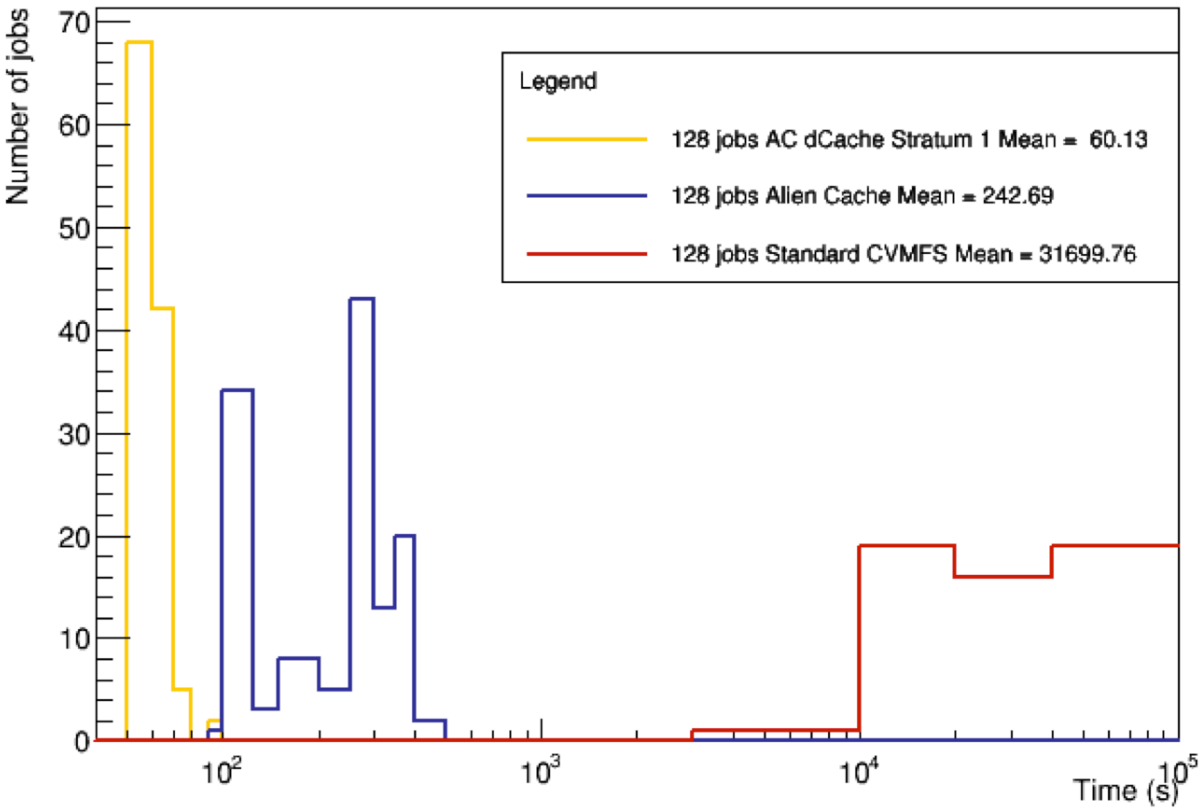

Figure 3. A histogram showing timings for jobs running the benchmark GENIE application

The x-axis in Figure 3 is the logarithmic time for a job to read the 2GB of auxiliary data, and the y-axis is the number of jobs within the histogram bin. All three trials were run on the same cluster at the University of Nebraska. The red line was ordinary CVMFS, reading through a particularly small HTTP proxy[2]. The blue line was using alien cache CVMFS on Lustre with an ordinary Stratum 1 (also in Nebraska); jobs took an average of 243 seconds to read the 2GB (about four minutes). The jobs plotted on the yellow line used the same alien cache but with a dCache-based Stratum 1 at Fermilab; these jobs read 2GB in an average of 60 seconds. Cache contents were reset before each run started.

[2] Note these jobs took an average of about 9 hours to read the 2GB, much worse than when we ran the benchmark at Fermilab as described in Section 2.

The results using dCache at Fermilab and HDFS at Nebraska as the alien cache are analogous to those shown in Figure 3. So far, we have scaled the tests to around 400 simultaneous jobs using Fermilab's dCache as the alien cache.

5. Implementation Details

Minor patches to the CVMFS client were needed for Lustre, HDFS, and dCache. The most significant changes were required for the dCache NFS v4.1 alien cache due to issues with stale file handles when temporary files were removed. The patches are released in CVMFS version 2.1.20.

The dCache-based Stratum 1 uses a virtual machine with the CVMFS server software installed. It is configured similarly to other Stratum 1s, except that the HTTP server forwards all data requests to a dCache-based WebDAV endpoint. No changes were required to the dCache WebDAV implementation; however, one final patch was needed to the CVMFS client to support the HTTP redirection. This allows the dCache WebDAV implementation to redirect HTTP requests to separate dCache HTTP servers running on many high-throughput data servers. In that way, the front end only serves a small volume of data.

The client configuration is also similar to the traditional CVMFS client configuration. Once the POSIX filesystem is made available on the worker nodes and an area made writable by the *cvmfs* user, the system administrator must create the file `/etc/cvmfs/domain.d/<domain>.local` with the following contents:

```
CVMFS_SERVER_URL="http://cvmfss1data.fnal.gov:8000/cvmfs/@fqrn@"
CVMFS_PUBLIC_KEY=/etc/cvmfs/keys/<domain>.pub
CVMFS_ALIEN_CACHE=/path/to/cache
CVMFS_HTTP_PROXY=DIRECT
CVMFS_QUOTA_LIMIT=-1
CVMFS_SHARED_CACHE=no
CVMFS_FOLLOW_REDIRECTS=yes
```

This configuration will enable mounting all repositories available in the domain. The `CVMFS_HTTP_PROXY` configuration disables the use of a proxy, and the `CVMFS_QUOTA_LIMIT` and `CVMFS_SHARED_CACHE` variables disable the local disk cache. `CVMFS_FOLLOW_REDIRECTS` is needed for the high-bandwidth Stratum 1, since support of HTTP redirects is disabled by default. All other options besides `CVMFS_ALIEN_CACHE` can be placed into a centrally distributed shared configuration.

When publishing, CVMFS has the ability to break up large files into smaller chunks of a configurable size; the default is 8MB. Since only larger files are expected for the auxiliary cache, we set the chunk size to 64MB.

6. Future Work

We plan to test the alien cache configuration on EOS-based storage, another storage element used at Fermilab. Additionally, to test scales beyond 400 jobs, we are working toward getting the dCache-based alien cache mounted on a large portion of a production cluster at Fermilab.

The CVMFS client does not manage the space in the alien cache as done with local disk caches. Cleaning up cache files is a built-in feature of dCache *scratch pools*, but automated cleanup is not commonly available for the other storage elements. We plan to implement a separate cache cleanup mechanism that can be run on any machine mounting alien cache.

Before alien cache is put in production, a second site Stratum 1 will be needed for reliability.

The Open Science Grid is working on a separate project, StashCache [5], to make auxiliary data available from about 10 strategically placed caches; smaller nearby sites will be able to use the caching infrastructure without dedicated hardware. The plan for that project is to transfer the data through the xrootd protocol, but we see advantages to also making data available to the smaller sites through CVMFS from the StashCache infrastructure. This configuration is not suitable for alien cache (due to high latency and cross-site security issues), but we plan to try to configure the StashCache xrootd installations to act as a reverse proxy for a high-bandwidth CVMFS Stratum 1 (the xrootd server implementation can additionally export data over HTTP). On a cache miss, the StashCache would forward HTTP requests to the real Stratum 1. Clients will be configured to use a StashCache Stratum 1 with a minimal local cache for catalogs and open files. We believe this can be done using an entirely centrally distributed shared configuration so no per-site configuration is needed.

7. Conclusions

In this paper, we demonstrate a high-bandwidth, distributed, centrally written (write once, read many, or WORM) POSIX filesystem. This setup uses hardware and software that is already available at most large particle physics computational sites. We find the performance of the alien cache CVMFS setup to have acceptable performance for the prototypical GENIE application. It is easy to configure and – as it supplies a POSIX interface – it is easy to use. It has the potential to be useful for a variety of applications when widely deployed.

Acknowledgements

Fermilab is operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy. This work is supported in part by the National Science Foundation through awards PHY-1148698.

References

[1] Blomer J, Buncic P, Charalampidis I, Harutyunyan A, Larsen D, and Meusel R 2012 Status and future perspectives of CernVM-FS *J. Phys.: Conf. Ser.* **396** 052013
[2] dCache: http://dcache.org/ *Last accessed on 14 May 2015*
[3] Lustre: http://lustre.org/ *Last accessed on 14 May 2015*
[4] Hadoop HDFS: https://wiki.apache.org/hadoop/MountableHDFS *Last accessed on 14 May 2015*
[5] StashCache: https://twiki.opensciencegrid.org/bin/view/SoftwareTeam/SW023_XrootdAcrossOsg *Last accessed on 14 May 2015*