

SCD Storage Access Architecture Draft

Motivation

- POSIX access to NFS mounted file systems from many worker nodes is very hard to support at scale → Would like to remove POSIX access to NFS mounted file systems from batch nodes and migrate all experiment's storage access to non-POSIX access to scalable storage systems
- Scale is only going to increase in the future, need a scalable architecture for the future
- New resource provisioning forms (HEPCloud: Clouds and HPC machines) will be integrated in the FNAL facilities in the future
- industry is pushing towards object stores and other technologies that move away from POSIX access

Goal

- Define storage access architecture that is scalable and maintainable in the future
 - removing POSIX access from worker nodes
 - removing mounted file systems on the worker nodes
 - consider only access to files stored in filesystems or filesystem-like areas
 - don't consider access to auxiliary blobs of data from databases (NoSQL or otherwise)

Process

- Step 0: preparation
 - Describe phase space: define categories for
 - Job types, Storage types, types of file inputs and outputs for jobs
 - Define which storage categories will be supported for which inputs/outputs of which job categories
 - Describe current system
- Step 1: adapt architecture to remove POSIX access from worker nodes
 - consolidate storage access for all job types
- Step 2: socialize architecture with experiments
 - gather feedback through CS-Liaison meeting presentation(s) and individual discussions with experiments
 - update the document using the feedback to make it more comprehensible for the community
 - discuss changes and timelines with the experiments
- Step 3: evolve architecture for the future

Job categories

- **Production** batch jobs (centrally submitted and managed, well defined codebase)
- **Analysis** batch jobs (user defined code)
- **Interactive** applications

Storage categories

- **tape-backed MSS**
 - *Mass Storage System (MSS) orchestrating disk servers and tape robots with tape drives*
 - *Provides access protocols to write files to disk, which are then automatically written to tape*
 - *Provides access protocols to read files from disk, which are automatically staged from tape if no disk replica exists*
 - *Naturally garbage collected, least accessed file replica on disk gets removed if space is needed*
- **non-tape-backed MSS**
 - *Mass Storage System (MSS) orchestrating disk servers*
 - *Provides access protocols to write files to disk*
 - *Provides access protocols to read files from disk*
 - *Two varieties:*
 - *Garbage collected, least accessed file replica on disk gets removed if space is needed*
 - *Persistent, when running out of space, writes fail*
- **OSG StashCache**
 - *Public Xrootd cache infrastructure on OSG*
- **network attached storage**
 - *Disk system providing full POSIX access*
- **local file system on worker nodes**
 - *Disk local to worker nodes usually accessible as scratch space*
- **CVMFS**
 - *Read-only distributed file system based on HTTP caches*
- **Sandbox**
 - *Group of files or tarball with files needed for the execution of a batch job*
- **Blob DB**
 - *DB infrastructure including REST APIs to retrieve stored blobs of data*

Protocol categories (as seen from job/application)

- **Copy In:**
 - copy files to local file system on worker node or interactive node
- **Copy Out:**
 - copy files from worker or interactive node to storage
- **Native streaming:**
 - access files through native protocol of storage
 - Example: dcap protocol for dCache
- **Xrootd:**
 - Access files through xrootd protocol (requires xrootd server infrastructure)
- **Submission infrastructure:**
 - Use file transport mechanism of submission infrastructure
- **HTTP:**
 - Download files through HTTP protocol
- **POSIX-like:**
 - Access files through POSIX-like interfaces to storage
 - Interface does not provide full POSIX functionality
 - Examples: dCache-NSF4, EOS-FUSE
- **POSIX read:**
 - Read-only access to files through fully POSIX compliant interface to storage
 - Example: accessing releases from CVMFS or reading files from network attached storage
- **POSIX write:**
 - Write access to fully POSIX compliant interface to storage
 - Example: writing files to network attached storage
- **Blob DB read:**
 - Access blobs from Blob DB through REST APIs

File input categories

- **code files:**
 - *run time executable, libraries and code files; two classes:*
 - immutable base releases of experiment code
 - user-specific code as add-on to base release or stand-alone
- **support files:**
 - *job specific inputs*
 - Examples: configuration files, txt files
 - < 10 MB per job

- **auxiliary files:**
 - additional inputs to processing and analysis jobs that have high reusability rates
 - Examples: flux files, pile-up files
 - 10 MB to N GB ($N \geq 1$)
- **data files:**
 - *holding data from data taking and MC simulation, low reusability rates*
 - Examples: RAW detector files
 - 100 MB to N GB ($N \geq 1$)

File output categories

- **log files:**
 - *text files holding status and error outputs produced during execution of applications, accessed for problem debugging*
 - Examples: stdout, stderr
- **histogram files:**
 - *Output generated by applications that can directly be used for end-analysis and is limited in size*
 - Example: histograms in root files, small ntuples in root files
 - 10 MB to 100 MB
- **output files:**
 - *holding data from data taking and MC simulation, low reusability rates*
 - Examples: RAW detector files
 - 100 MB to N GB ($N \geq 1$)

Current system

MSS is provided through dCache and Enstore. dCache provides a front-end for enstore (tape backed) and can also provide non-tape backed cache or persistent disk storage. All access to MSS is through dCache using either dCache protocols (dcap, gsiftp, nfs v4.1, ...) or SAM/IFDH tools.

Enstore also provides a Small File Aggregation (SFS) feature that is heavily used. Writing small files to tape is very inefficient. SFA aggregates small files on disk according to aggregation policies.

Interactive disk is provided through BlueArc NAS, and code distribution through CVMFS.

- **tape-backed MSS:**

- **Shared Production:** Enstore (tape) and dCache (disk cache front-end). Access through dCache protocols or SAM/IFDH. Implemented as dCache “production” rpools with a file lifetime of about 100 days. Intended use is for production.
- **Per experiment write-only:** These pools are write only. If a file in these pools are referenced for read, the file is first copied to production space for access. This is a write-through cache intended for streaming files to tape that won’t be accessed in the immediate future.
- **non-tape-backed MSS:**
 - **Shared cached “Scratch” :** Implemented as dCache “Scratch pools” with a file lifetime of 30-60 days. This cache is intended for short-term storage of files.
 - **Per experiment non-Cached “persistent”:** Implemented as dCache persistent pools. Files are not automatically evicted. If the space fills, new writes will result in errors until users remove files to make more space. This space is intended for analysis use.
- **Network Attached Storage:**
 - BluArc NAS. This network file system is intended for interactive use and provides POSIX I/O. Analysis workflows have been performed using BlueArc disk but it does not perform well for this purpose. A goal is to move storage for analysis processing to dCache “persistent” or other suitable storage.
- **Code Base:**
 - CVMFS

Input file support for production batch jobs

- **code:**
 - base releases: CVMFS
 - user-specific code: sandbox through submission infrastructure or HTTP
 - copy-in from non-tape-backed MSS
- **support:**
 - sandbox through submission infrastructure or http
 - Blob DB read
 - Copy In from non-tape-backed MSS
- **auxiliary:**
 - Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution
 - Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution
 - Xrootd access to non-tape-backed MSS or OSG StashCache or tape-backed MSS if going through DM solution
- **data:**

- Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution
- Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution
- Xrootd access to non-tape-backed MSS or tape-backed MSS if going through DM solution

Output file support for production batch jobs

- **log:**
 - transport through job submission infrastructure
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
- **histogram:**
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
- **output:**
 - Copy out to non-tape-backed MSS or tape-backed MSS if going through DM solution

Input file support for analysis batch jobs

- **code:**
 - base releases: CVMFS
 - user-specific code: sandbox through submission infrastructure or HTTP
 - copy-in from non-tape-backed MSS
- **support:**
 - sandbox through submission infrastructure or http
 - Blob DB read
 - Copy In from non-tape-backed MSS
- **auxiliary:**
 - Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution
 - Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution
 - Xrootd access to non-tape-backed MSS or OSG StashCache or tape-backed MSS if going through DM solution
- **data:**
 - Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution

- Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution
- Xrootd access to non-tape-backed MSS or tape-backed MSS if going through DM solution
- remark: if production jobs support custom code, analysis and production jobs are the same concerning storage access
 - only difference: authentication
 - Production jobs: group accounts or VOMS roles for reading and writing
 - Analysis: user accounts or credentials, need to be mapped properly

Output file support for analysis batch jobs

- **log:**
 - transport through job submission infrastructure
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
- **histogram:**
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
- **output:**
 - Copy out to non-tape-backed MSS or tape-backed MSS if going through DM solution

Input file support for interactive applications

- **code:**
 - base releases: CVMFS
 - user-specific code: sandbox through submission infrastructure or HTTP or POSIX read access to network attached storage
- **support:**
 - sandbox through submission infrastructure or http
 - Blob DB read
 - POSIX read access to network attached storage
 - Copy In from non-tape-backed MSS
- **auxiliary:**
 - Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution
 - Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution

- Xrootd access to non-tape-backed MSS or OSG StashCache or tape-backed MSS if going through DM solution
- POSIX read access to network attached storage
- **data:**
 - Copy In from non-tape-backed MSS or tape-backed MSS if going through Data Management (DM) solution
 - Native streaming from non-tape-backed MSS or tape-backed MSS if going through DM solution
 - Xrootd access to non-tape-backed MSS or tape-backed MSS if going through DM solution
 - POSIX read access to network attached storage

Output file support for interactive applications

- **log:**
 - transport through job submission infrastructure
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
 - POSIX write to network attached storage
- **histogram:**
 - Copy out to non-tape-backed MSS
 - DM solution provides aggregation and archival functionality to tape-backed MSS
 - POSIX write to network attached storage
- **output:**
 - Copy out to non-tape-backed MSS or tape-backed MSS if going through DM solution
 - POSIX write to network attached storage

Guiding principles for community

- File organization is handled by the experiment
 - Aggregation to reasonable sizes is expected
 - SFA is last effort to avoid overloading the tape system
- Logfile and histograms need to be aggregated before going to tape
- Anything going to tape need to go through a data management (DM) solution (SAM, PhEEx, ...)

Changes for community

- Base releases are only accessible from CVMFS (for compatibility with HEPCloud)
- No POSIX access to any storage for batch jobs
- No POSIX-like mounts to EOS and/or dCache from interactive nodes

Questions

- Should we define rough target time scales to migrate current experiments to this architecture?
- Should we define guidelines for new experiments to be compatible with this architecture

R&D areas

- non-tape-backed MSS systems
- storage access from HPC installations
 - cvmfs vs. /& docker
 - ...
- Consider what happens when we move to storage based on addressable objects (merge database with current storage technology)?
- Enstore vs. HPSS vs. other tape systems
- Steve Timm: limited flexibility in the mid-range storage
 - small persistent servers of scale of 1-20 TB that are serving or handling things other than big experimental data. This includes much of the smaller scientific database stuff, a number of services within grid services (Gratia for instance), servers that collect a large amount of log files, etc.
 - The current virtualization methods in play (CCD's vmware setup, the new GPCF2 setup, and FermiCloud) don't have a good way to address this. FermiCloud could and used to do this, but found it to be a difficult administration task. GPCF2 could do it if the remaining storage issues in the RHEV setup are set.
 - Ideally you would like to have a setup where persistent data doesn't exist on local disks of a single server and can be easily migrated to other servers...but this requires either NAS such as Bluearc (FermiCloud's fallback position) or a reliable clustered file system on a SAN (which does not exist).
 - CERN has had good success with Ceph but it requires basically a full FTE to maintain the system and also running a much more bleeding-edge OS version and associated libraries than Fermilab has historically been comfortable doing.
 - The fallback position has been a class of miscellaneous bare metal servers that are badly over provisioned for what they do, poorly backed up if at all, and not

flexible to move from one machine to the other. This is wasting significant effort on the part of admins, money in the hardware budget, and costing reliability to the users. We need a better and more flexible infrastructure.

- Use cases:
 - i. Unique or novel analysis cases requiring continued access to much more data and/or database (and RAM) than normal. the Nova LEM case is one example. There are several DES cases. There will probably be others as we go along.
 - ii. Mid-range databases. Several of them exist for test purposes on FermiCloud at the moment. DES has one such machine right now. Yuyi Guo and several others in her group have played with couchDB or MongoDB and friends. I am not sure if these are used by end-users or by service maintainers but it's a growing and popular FermiCloud use case and sooner or later these people will want to go to production.
 - iii. Databases and large log file caches for services. Gratia is one example (having 1TB for log file caches in addition to 4-5 TB of actual database). This system at one point was virtualized but Ed Simmonds insisted on buying separate bare metal machines for databases and collectors due to the problem of not wanting to attach large persistent data to a virtual machine. As a result we have replaced what was 4 bare metal machines with 13.
 - iv. The build service.. "Jenkins" cries out for a cloud-based system with virtual storage.. In fact it is possible to buy Jenkins-as-a-service on Amazon Web Services and many companies do...it is their preferred way of delivering their product.

Documents

- "Recommendations for dCache, IFDH and SAM tools" for NOvA from 28. July 2015
 - <https://drive.google.com/open?id=0B-phn6w6XemFhN1p2SVRWbVE>
- "FIFE Architecture Design Report"
 - <http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=5180>

Glossary

Storage Access Architecture Matrix

	Production		Analysis		Interactive	
	Input	Output	Input	Output	Input	Output
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	output	auxiliary data	output
non-tape-backed MSS	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output	code support auxiliary data	log histogram output
OSG StashCash network attached storage	auxiliary		auxiliary		auxiliary	
local file system on worker nodes					code support auxiliary data	log histogram output
CVMFS	code		code		code	
Sandbox	code support		code support			
Blob DB	support		support		support	

	As seen from Production/Analysis Batch Jobs									
	Copy In	Copy Out	Native Streaming	Xrootd	submission Infrastructure	HTTP	POSIX-like (dCache-NFS4, EOS-FUSE)	POSIX read	POSIX write	Blob DB read
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	auxiliary data						
non-tape-backed MSS	code support auxiliary data	log histogram output	auxiliary data	auxiliary data						
OSG StashCash network attached storage				auxiliary						
local file system on worker nodes										
CVMFS								code		
Sandbox					code support	code support				
Blob DB										support

	As seen from Interactive Applications									
	Copy In	Copy Out	Native Streaming	Xrootd	submission Infrastructure	HTTP	POSIX-like (dCache-NFS4, EOS-FUSE)	POSIX read	POSIX write	Blob DB read
tape-backed MSS (going through DM solution)	auxiliary data	output	auxiliary data	auxiliary data						
non-tape-backed MSS	code support auxiliary data	log histogram output	auxiliary data	auxiliary data						
OSG StashCash network attached storage				auxiliary				code support auxiliary data	log histogram output	
local file system on worker nodes										
CVMFS								code		
Sandbox					code support	code support				
Blob DB										support

tables from:

https://docs.google.com/spreadsheets/d/1ZsIn2ZXbgYBSBI_TD6vT5db4FLUfxK0cu-m4zFK8myw/edit?usp=sharing

URL to document

<https://docs.google.com/document/d/1EwvmQ79IR6zB8AyHbSfKHM55RrpQzNWyt-H1Gbkf46I/edit?usp=sharing>

DocDB entry: CS-doc-5606