Taking Global Scale Data Handling to the Fermilab Intensity Frontier

Adam L. Lyon, Robert A. Illingworth, Marc Mengel, Andrew J. Norman

Data Handling Group, Running Experiments Department
Scientific Computing Division
Fermi National Accelerator Laboratory
PO Box 500, Batavia, IL 60510-0500 USA

lyon@fnal.gov

Abstract. SAM is a comprehensive data management system used by the Tevatron Run II experiments with great success. The newest experiments at Fermilab, the Intensity Frontier experiments, are currently lacking such a system. In these proceedings, the advantages of using SAM for these experiments is discussed. Two improvements to SAM, namely SAMWeb and SAMfs are described. These improvements will make SAM much easier to integrate, deploy, maintain, and use.

1. Introduction

The Fermilab Intensity Frontier (IF) experiments involve precision measurements of some of the most interesting phenomena in particle physics including neutrino oscillations, muon to electron conversion, and the muon anomalous magnetic moment. Several experiments are ready to run, like MINOS+ and Minerva. NOvA and MicroBoone are in the construction phase. The Mu2e and new Muon g-2 experiments along with LBNE are planned for the near future. All of these experiments generate large amounts of data, whether collected by the apparatus itself or simulated for studies and predictions.

The current storage philosophy followed by these experiments has been to store as much data as possible on network attached storage. This storage system (the "BlueArc") is accessible to many computing nodes at Fermilab, including worker nodes running in the "FermiGrid" batch environment[1]. While convenient, this paradigm has lead to several problems,

- Capacity planning is difficult and many experiments frequently reach their allotted space and need to request more.
- Organization is only by directory trees. It is difficult to link data files with metadata. Discovering data files and making datasets is inconvenient.
- There are no user-accessible tools for archiving data on tape and then retrieving those files again later.
- While the BlueArc storage system is accessible widely at Fermilab, there is no easy way to utilize offsite resources and have access to the data.
- Scalability is an issue as the facility space and resources fill up faster for disk than for tape.

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

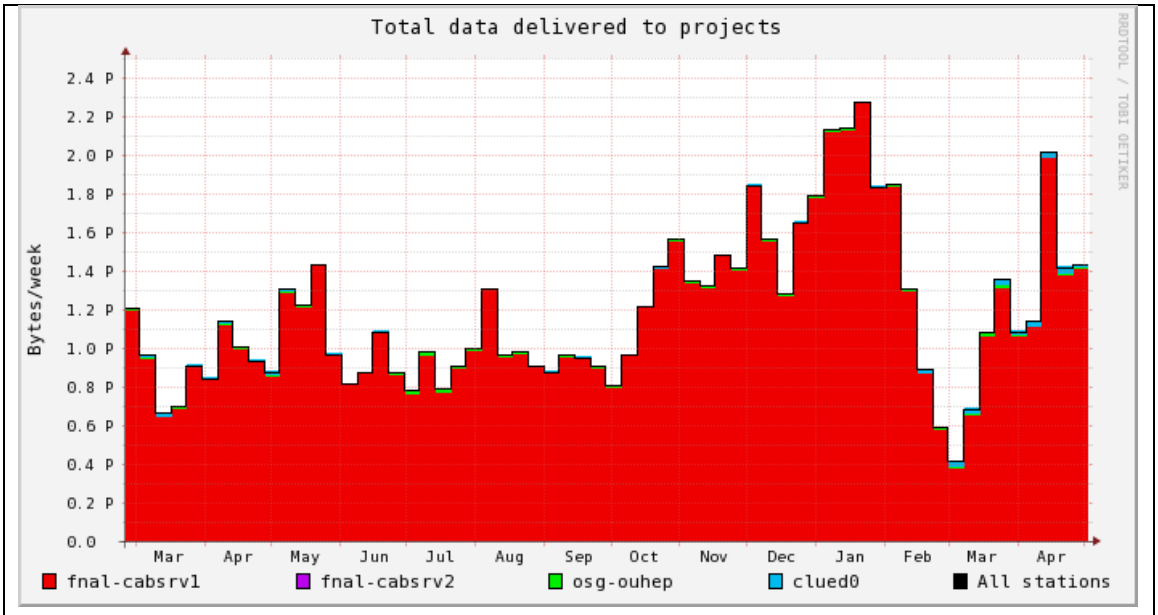

Figure 1. Weekly delivery rates of files with SAM in Petabytes per week for the D0 experiment from February 2011 through April 2012.

In contrast, the Tevatron Run II experiments, CDF and D0, ran with a comprehensive data management and delivery system written by the Fermilab Computing Division called SAM (Sequential Access via Metadata). SAM provides these capabilities,

- Catalogs data files with metadata. Datasets are constructed from the results of metadata queries.
- Maintains a replica catalog so that the cached locations of files are known.
- Delivers data on demand from archival storage to caches located world-wide (caches are automatically managed – continued use of popular files keeps them in the cache; files that are no longer uses get replaced by more popular files).
- Closely integrates with the Fermilab tape archive system (Enstore[2]).
- Closely integrates with other data handling systems. For example, SAM can utilize dCache pools for cache or maintain its own cache disks. SAM can utilize grid storage elements with SRM and gridFTP tools.
- SAM follows a client–server model and provides multiple interfaces for access and administration including a command line interface (CLI), a python client, and a C++ client. The clients communicate to the servers via CORBA.

SAM has been running for nearly ten years at the Run II experiments and has been extremely successful. It is now mostly in maintenance mode and requires little human intervention to keep running. It routinely delivers over 300 TB of data per day to each experiment, as indicated in Fig. 1. A coarse component diagram is shown in Fig. 2. Given its success, it is natural to attempt to bring SAM to the Fermilab IF experiments that are in need of a comprehensive data management and delivery system.

2. SAM at the Intensity Frontier Experiments

SAM addresses many of the deficiencies of the IF experiments current storage model. SAM will organize data by metadata and allow for easily created datasets. It can use Bluearc space as a cache

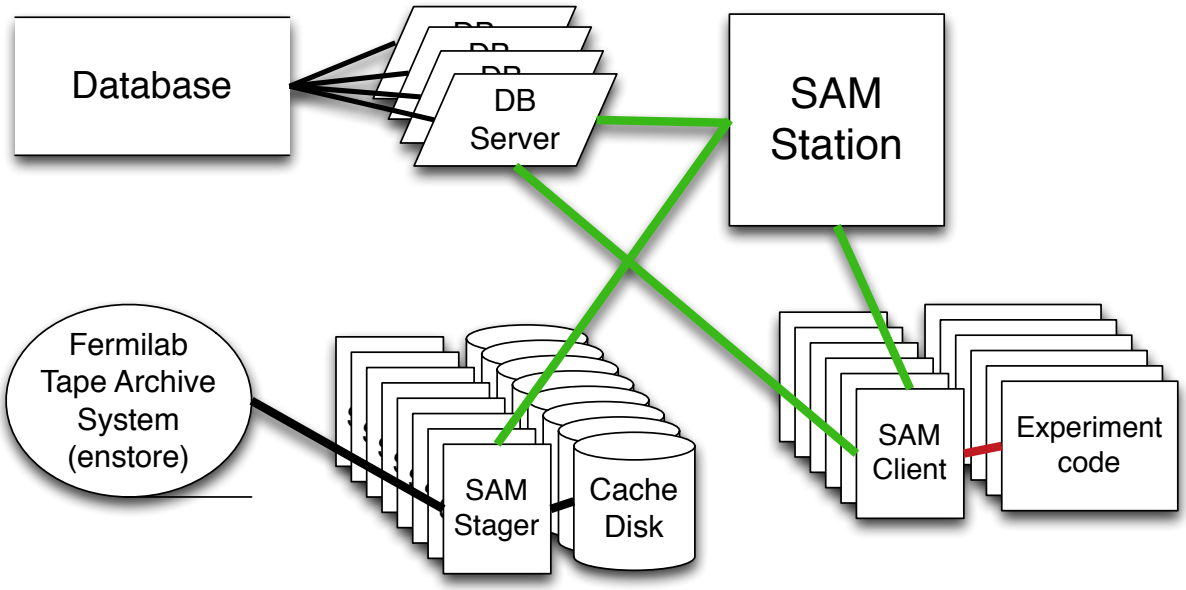

Figure 2. A coarse component diagram of SAM. Green lines indicate communication via CORBA. The SAM station orchestrates all aspects of file delivery and caching. SAM stagers are responsible for moving data to and from cache disks. A database holds the metadata and replica catalogs and communicates with the outside world via DB servers that contain business logic and queues for scalable access. Experiment code accesses SAM functions via the SAM specific C++ API (red line) calls to the SAM client. The SAM client communicates with the SAM station and DB servers to perform actions and requests.

and will manage the files there automatically, eliminating the storage space overruns. SAM is designed to deliver files to offsite locations, and so running on remote sites will now be possible. There is, however, some work to accomplish to make SAM more adoptable by the IF Experiments.

2.1. Improving the Framework Integration – **SAMWeb**

Both of the Run II experiments, CDF and D0, wrote their own complex reconstruction/analysis framework systems in C++. SAM provides a C++ client with an API for various actions, like registration of a job with the SAM system, specifying the dataset to use, requesting a file, releasing a file, and declaring the job a success or failure. Integrating this SAM component into the experiments' framework was a very complicated and time consuming task and required a SAM developer to have expert knowledge of the framework as well as an experiment developer having expert knowledge of the SAM C++ API. Furthermore, to use the C++ API, the SAM C++ client must be installed on each node running a framework application. These connections are shown in detail in Fig. 3. Making updates to the SAM client would involve deploying the new version to a large number of machines, including ones with remote access to SAM. This situation was a logistical headache.

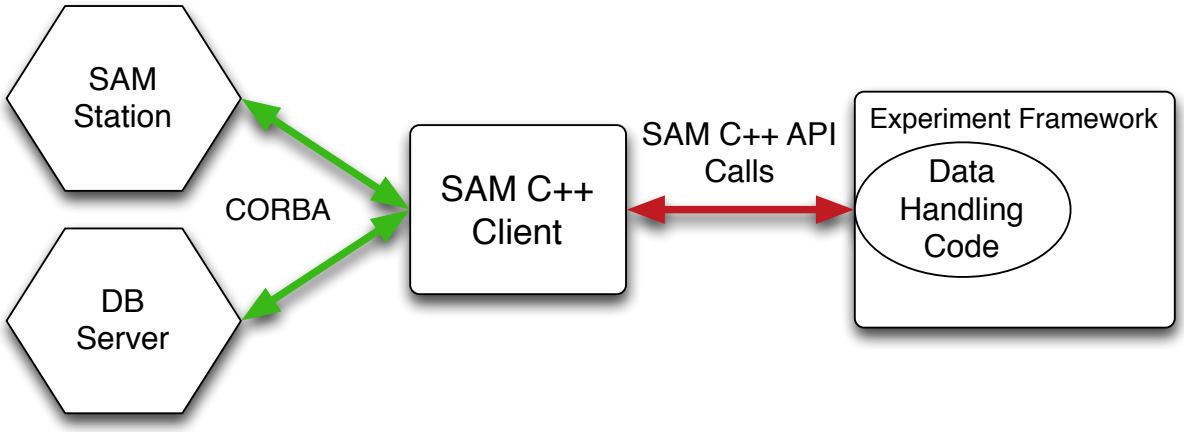

Figure 3. Detail of the traditional connections between SAM and experiment code.

For the IF experiments, there are three main frameworks in use. Minerva's framework is based on Gaudi. MINOS+ uses a Root based framework written for the original MINOS experiment. The other experiments, NOvA, LBNE, Mu2e, and g-2, are using Art[3], a common framework written by the Fermilab Scientific Computing Division. Integrating with three frameworks similar to what was done for CDF and D0 would be a tall order.

SAMWeb is introduced to alleviate the difficulties in framework integration and deploying the SAM client. A detail of the communication is shown in Fig. 4. The major difference is that instead of integrating a SAM API piece into the experiment framework for communicating with SAM, communication is performed via an http protocol. The SAMWeb server converts the http requests into SAM requests with the SAM python API (the python API is used because it is simpler than the C++ API) and returns the results back to the framework via http. While this scheme looks more complicated than the traditional system, it in fact has some very desirable advantages,

- There is no SAM specific code in the framework, and it is not directly coupled to SAM (e.g. it loads no SAM library). SAM could be replaced with some other data management system and the experiment framework and data handling code need not change.
- Writing the data handling code is much simpler, because intimate knowledge of SAM is not needed. The code need only know how to make certain http requests and deal with responses.
- SAM is completely experiment agnostic. No knowledge of the experiment framework is necessary.
- If the SAMWeb server needs to be scaled up, more instances of it can be generated.
- The SAM client is only deployed with the SAMWeb server. Deployment to the experiment is unnecessary. The SAM client can be updated easily without touching the experiment code.
- Deploying to remote sites is easy as the SAMWeb servers remain at Fermilab. There is nothing to deploy at remote sites.

The decoupling of SAM from the experiment code is a great feature and greatly simplifies deployment and updates of the SAM client. Note that SAMWeb does not contain all of the functionality of the SAM client, but instead has the necessary functions for supplying an interactive or batch job with files. Many administrative tasks will still require the full SAM client, but that client would then need only be installed on a very small number of machines (perhaps just one central one). In fact administrative actions that become popular could be added to SAMWeb, eliminating the need for deploying the SAM client at the experiment.

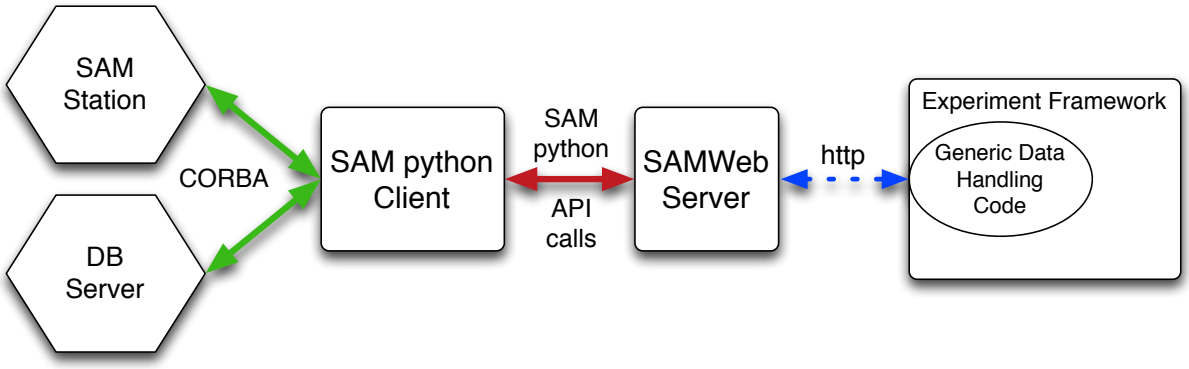

Figure 4. Detail of connections between experiment code and SAM with SAMWeb.

New functionality could also be added beyond what the SAM client already contains. For instance, one could add functions to the SAMWeb server that communicate directly with the DB server or database itself. Experiment specific functionality can also be more easily accommodated with this system. Two such additions are "Dataset Definition Editors" (DDEs), one for Minerva and another

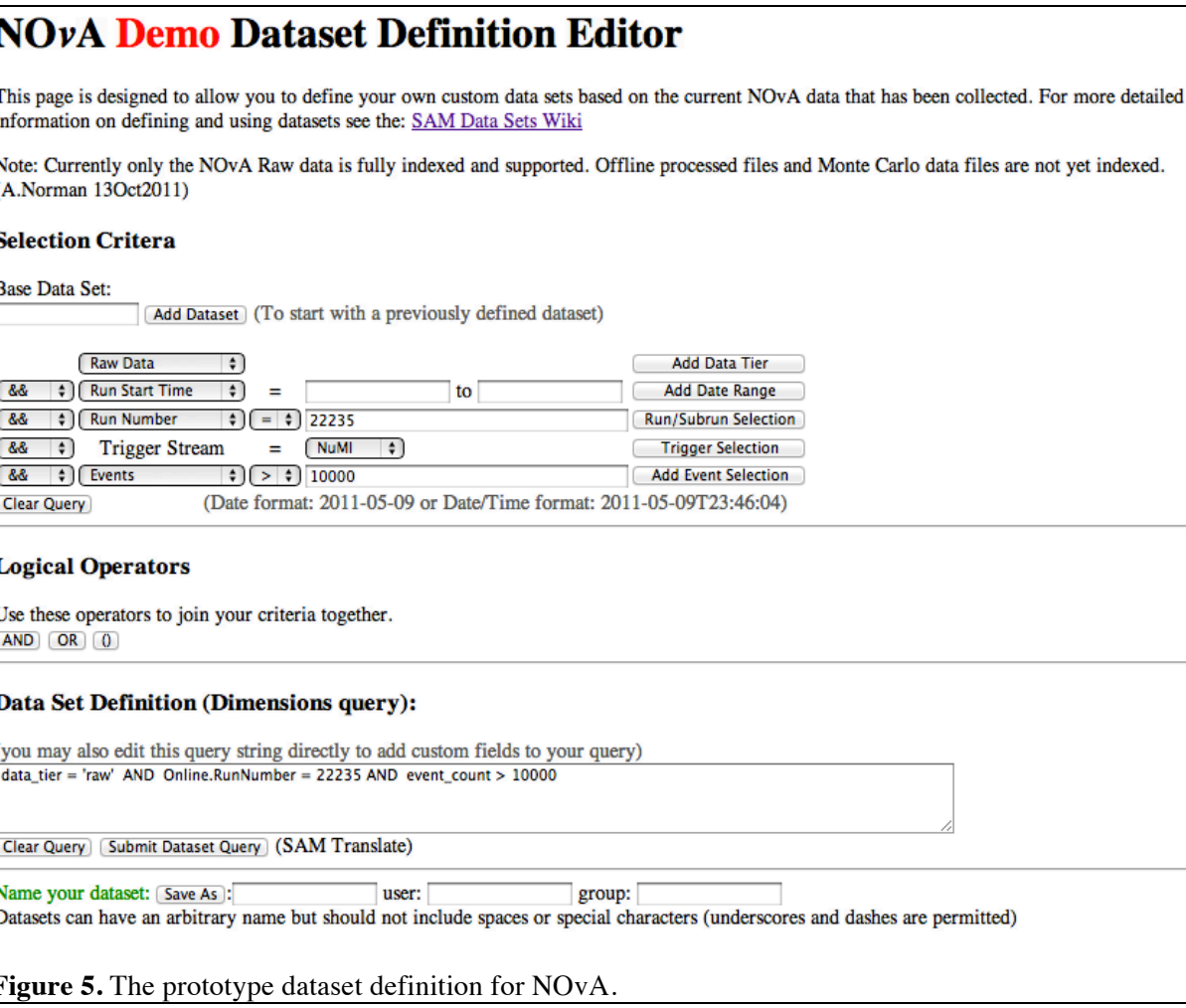

Figure 5. The prototype dataset definition for NOvA.

for NOvA are new parts of SAMWeb. A DDE is a web application that allows users to make guided metadata queries to create datasets. An example of a prototype for NOvA is shown in Fig. 5.

A first version of SAMWeb is nearly complete, and the process of integration into the experiments' code has begun for Minerva and NOvA. Minerva uses Gaudi as its framework. Integrating the traditional SAM client would have been extremely difficult, and having Gaudi interact with an http server is proving difficult as well. As a temporary work around, a Minerva job will launch a python script that will send the http requests to retrieve a file for the job. The python script will then launch Gaudi and have it run over that file. Native integration with Gaudi is under study. For NOvA, the ART developers are adding the http functionality to their framework.

2.2. Improving the interactive SAM experience – **SAMfs**

A lesson learned from Run II is that the SAM interactive experience for regular users requires improvements. On a Linux system, when a user wants to deal with files he or she typically uses the shell's file system commands (`cd`, `ls`, `cp`, `cat`). SAM has no such interface and its alternative is difficult to learn and remember.

```
sam translate constraints –dim="run_number 155934 and data_tier reconstructed"
```

Listing 1. SAM CLI command for listing files with a particular run number and data tier.

Files in SAM are cataloged by metadata. A user can obtain a list of files with the metadata query command, `sam translate constraints`, as shown in Listing 1. In that example, the user requests a listing of files that satisfy the query of a particular run number and data tier. The data tier is related to the processing level of the data within the file. For example, unprocessed data collected by the detector have a data tier of "raw". Data processed by the reconstruction program have a data tier of "reconstructed". Instead of forcing users to write SQL queries, a simplified domain specific language was written for SAM called "dimensions". With dimensions, users can construct fairly complicated queries using a rich set of fields (e.g. `run_number`). However, our experience from Run II indicated that the vast majority of users have no need for complex queries and in fact most run small variations of a limited set of simple queries. The guided SAM dataset definition editors, an example of the NOvA instance is shown in Fig. 5, are easier to use than the command line tool, and in preparing for batch processing they are the tool of choice. But this situation is still complicated for users that want to run interactive applications.

While most processing is performed in batch jobs, running one or two files through an application interactively is important, especially for testing and debugging. Retrieving files to a local disk from SAM for interactive use is quite difficult with the current SAM toolset. An idea, not fully developed yet, is to add a file system-like interface to SAM. As mentioned above, when dealing with files the user utilizes the shell file system commands. Why should SAM be any different? This idea is named SAMfs.

The SAMfs interface is made possible by Filesystem in Userspace (FUSE) [4]. FUSE implements a fully functional file system interface for a given application. For a particular mount point, file system calls are routed to FUSE which passes those calls to user written interface code. In SAMfs, a limited but useful interface to SAM will be presented as a file system. Some use cases and examples are listed below.

```
> ls -F /samfs/run
ERROR-supply-a-run-number-like-200000@

> ls -F /samfs/run/268211
111/   112/   113/

> ls -F /samfs/run/268211/111
raw/   reconstructed/

> ls -F /samfs/run/268211/111/raw
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root* raw_nu_268211_111_3.root*
```

Listing 2. SAMfs navigation to files via metadata directory trees. The first command simply asks for all runs. Such an action results in a very expensive query and a huge return list, therefore an "error" is returned asking the user to give an actual run number. This error is actually a fake file name returned through FUSE. The second command performs a query given a particular run number and returns a list of available sub-runs (sub-runs are partitions of a run). Note that the sub-runs are represented by directories, indicating that there are more levels of the tree to dive into. The third command specifies a sub-run and then a list of data-tiers are returned. The final command fills in all of the necessary information and returns a list of actual data file names. The use of `-F` is discussed in the text.

2.2.1. File listings via metadata

SAMfs will allow a user to navigate a limited set of metadata fields in the form of directory trees with SAMfs performing the actual SAM metadata query behind the scenes. An example is shown in Listing 2. Note that in the examples the standard command "`ls -F`" is used. The option means that the file type is indicated by a symbol: slashes for directories, * for regular files and @ for symbolic links. Files that are not cached are listed as regular files. Files that are in the cache are listed as symbolic links. Listing cached files as symbolic links is advantageous since upon access, FUSE simply returns the actual location of the file in the cache and the operating system will stream it to the application, relieving SAMfs of that task better suited to the OS. Another example in Listing 3 shows navigating pre-defined datasets created by a user (datasets are defined by the regular SAM command line tool or a DDE).

```
> ls -F /samfs/dataset
ERROR-supply-a-user-name@

> ls -F /samfs/dataset/lyon
my-special-run-15593/  oscillated-nu-bars-5/  bad-tracking-examples/

> ls -F /samfs/dataset/lyon/my-special-run-15593/
reconstructed_nubar_15593_112_4.root*  reconstructed_nubar_15593_112_5.root*
```

Listing 3. SAMfs navigation to files via metadata directory trees, similar to listing 2, but this time navigating through pre-defined datasets.

2.2.2. Opening a file via SAMfs

Opening a file not in the cache means that SAM must move the file from a different cache or archival storage. SAMfs makes this work transparent to the user and is only noticeable as a time delay while the file is being moved. Once SAM has moved the file to the machine's designated SAM cache, the actual location of the file is returned via FUSE. Since FUSE tags the file as a symbolic link, the OS will open the actual file for the application.

There is a complication with this scheme. FUSE does not know the difference between dereferencing a symbolic link for an `ls` command or for open. Retrieving a file not in the cache is a potentially very expensive operation and without some protection, performing an `ls` could inadvertently trigger a large number of unwanted file transfers. Furthermore, using wildcards would create a similar problem. SAMfs is meant for opening a small number of files interactively, and so a protection scheme could involve adding a special "directory" in the file path called `get`. When SAMfs sees a path with `get` it will give an error unless a full file path is specified (no wildcards). An example of this behavior is shown in Listing 4. If used correctly, SAM will move the corresponding file into the cache and it will be opened by the application. If the file is already in the cache, its actual location is simply returned to the OS and it is opened immediately. Listing 4 also depicts giving a SAMfs path within an application such as root.

```
> ls -F /samfs/run/268211/111/raw
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root* raw_nu_268211_111_3.root*

> ls -F /samfs/get/run/268211/111/raw
ERROR-trying-to-list-directory-with-a-GET-path@

> cat  /samfs/get/run/268211/111/raw/raw_nu_268211_111_2.root
# binary output here

> root
TFile* f = new TFile("/samfs/get/run/268211/111/raw/raw_nu_268211_111_2.root")
# ... back to shell

> ls -F /samfs/run/268211/111/raw
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root@ raw_nu_268211_111_3.root*
```

Listing 4. Examples of opening a file with SAMfs. The first command shows that all of the files are uncached (note the *). The second command tries the directory listing with `get` in the path, resulting in an error. The third command opens the file with `cat`. Since the file is not in the cache, SAM will do a file transfer from a different cache or the archive. The user may notice a significant delay. The user could also open the file directly in an application, such as Root, as depicted in the fourth command. Finally, performing a directory listing shows that the file is cached (note the @).

2.2.3. Caveats to SAMfs

There are several caveats to this SAMfs system that come from the limited interface the file system and FUSE provide. Some errors can be handled by returning fake files with descriptive names. In other instances, the only error that can be given is "No such file or directory". Such errors may be hard to interpret by the user. As mentioned, SAMfs is meant for navigation of very limited metadata fields. Certainly the regular SAM CLI and DDEs will offer a much richer set of queries. SAMfs is also only suitable for accessing a small number of files and will not trigger prefetching, so it will not be as efficient as the standard mechanism of accessing files that batch jobs employ. Many SAM operations do not easily lend themselves to the file system metaphor and so care must be taken to not "bend" the interface too much. But for the use case of opening a small number of files interactively for testing and debugging, SAMfs will be an important and welcome solution. In its implementation, SAMfs will most likely use SAMWeb for communication with the SAM services.

3. Conclusions

SAM is being introduced to the Fermilab Intensity Frontier experiments to provide comprehensive data management services. It will solve many of the organization and delivery problems these experiments face today. Most importantly, SAM will allow for effective use of storage space, easy access to archival storage, and makes remote access to the data possible. There are two aspects of SAM undergoing improvements to make it more acceptable to experimenters and developers. SAMWeb is an http based interface to the SAM client that eliminates direct coupling between SAM and the experiment's framework code. SAMWeb will greatly simplify the integration of SAM into the experiment's framework and will make the SAM client trivial to deploy and much easier to maintain. SAMfs is a limited file system like interface to SAM that is to be used when such an interface is most appropriate, that is when a user needs a small number of files for testing or debugging an application. SAM will bring global scale data handling to the IF experiments, and its alterations of SAMWeb and SAMfs will make SAM much easier to integrate, deploy, maintain and use.

The Fermi National Accelerator Laboratory is operated by Fermi Research Alliance, LLC under Contract No. De-AC02-07CH11359 with the United States Department of Energy. This contribution is FERMILAB-CONF-12-189-CD.

References

[1] S. Timm, et., al., "FermiGrid, A campus grid supporting local and remote users sharing resources across Fermilab and the Open Science Grid", in these proceedings

[2] G. Oleynik, et. al., Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2005)

[3] W. Brown, et. al., "The art framework", in these proceedings

[4] http://fuse.sourceforge.net