Interface between the SAM data handling system and CDF experiment software

V. Bartsch[1], A. Baranovski[2], D. Benjamin[3], K. Genser[2], S. Hsu[3],
E. Lipeles[4], R. St. Denis[5], I. Sfiligoi[2], S. Stonjek[6]
[1]University College London, [2]Fermi National Accelerator Laboratory, [3]Duke University,
[4]University of California, [5]University of Glasgow, [6]Oxford University

Abstract

The experiment CDF at Fermilab has recently changed its data handling system. The old data handling system called DFC (Data File Catalog) is working locally at Fermilab but it is impossible to distribute it. As a new data handling system SAM [1] was chosen which has been developed and used by DZero in 2000. SAM provides mechanisms to deliver and cache data. It is easy to build GRID extensions which manage the job handling, monitoring and brokering on top of the SAM data handling system.

In order to reduce the costs of the migration of the data handling system SAM needs to provide the same functionalities as DFC so that the users gain the same user experience. This requirement has mainly influenced the design of the interface between the CDF software framework and SAM. The change of the underlying data handling system is mainly transparent for the users.

In this article we present the requirements on the interface between the CDF data handling framework and SAM, the resulting design of the interfaces and their implementation.

CDF'S REQUIREMENTS

In the following main features of CDF analysis jobs and the resulting requirements on the interface to the data handling system are described:

- The data handling system and the interface have to scale under the full load of the users at CDF. Typically 500 SAM job sections run simultaneously on the worker nodes of the central analysis farms. The scaling of the SAM data handling system has been studied elsewhere [2].
- In order to increase the user's acceptance of SAM any change of the underlying data handling system should be completely transparent to the users and ideally give the user the choice between the data handling systems.
- A typical CDF analysis job runs several days and consumes several Tbytes of input data which translates into large file lists. Therefore the user jobs have to run reliably. In case of any error during execution a book keeping mechanism needs to be supported by the interface to the data handling system. The implementation of the automatic recovery of analysis jobs is described elsewhere [3].
- For the software development it is desireable to develop new software in independent packages so that the packages can be updated independently from each other and the development and testing cycles are not depending on the main part of the CDF software. Ideally a forward compatibility between the CDF framework and the interface to the data handling system should be achieved.

CDF FRAMEWORK

CDF analysis jobs are steered by a script which specifies among others the environment in which the jobs should run, several physics modules, debugging information, input and output files. The input and output modules are the preferred interfaces to access input data and direct output data in a standard analysis job. The input module invokes the data handling system in order to manage the data delivery to the job.

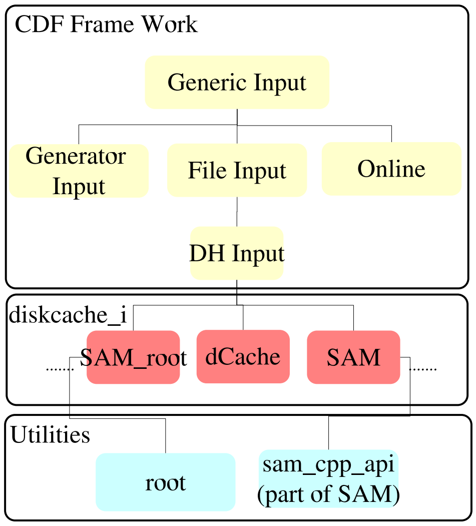

Figure 1: *Architecture of the software framework*

Fig. 1 shows an inheritance diagram of the part of the CDF framework which is relevant for this article. The CDF framework handles data from Monte Carlo production, from the data handling system and raw data. The DH input class is the interface to the data handling system.

INTERFACE OF TO THE DATA HANDLING SYSTEM

The DH Input class calls the interface to the data handling systems called diskcache_i (see Fig. 1). diskcache_i contains several shared libraries which can be called independently from each other by a generic stable interface. This gives the flexibility to implement accessors to different data handling systems like SAM or DFC. It allows forward compatibility of the CDF framework to diskcache_i versions. This feature is helpful for the development cycles. Any changes and improvements in either package can be immediately used independent of an update of the other code components. That means that even if an old CDF framework version is needed for reasons beyond the data handling the newest diskcache_i version can be loaded.

In order to address the scalability of the approach at the analysis farms diskcache_i does not call the SAM command line interface directly but calls an API which does not overload the code server.

SAM monitors the flow of a job with the help of a connected SAM project. The SAM project runs over the files of a dataset. A dataset is a collection of data which are being matched by the same metadata. The SAM project translates the dataset into a collection of files at the start of the SAM project. The SAM project needs to be started in the beginning of the job. The SAM database monitors the file delivery to the SAM project. Files are delivered to the project until either all files belonging to the specified dataset are delivered or a maximum number of files which has been specified is reached. At the end of the job the SAM project is closed and the job is terminated.

The diskcache_i software package has been tested thoroughly to ensure reliability for user jobs. During these tests several failure modes have been identified. File delivery problems need to be handled correctly. The correct start and end of the SAM project can be a failure mode, e.g. if a job is started after the SAM project has already finished or if the SAM project is not starting correctly. It was also ensured that there is no possibility to run in a mixed mode between the old and the new data handling system by creating scripts which are not strictly addressing one data handling system. A large number of tests have been run in order to catch any failure which have not been found in first place and to check reliability with the new data handling system. In the mode using DFC this software has been running since a long time, the changes made to work with SAM have been made last summer and it has been running since then without much maintenance.

An interface to SAM within root jobs is provided. It returns the files delivered by SAM, so it can be used in a sequential loop over the files defined in a project. A typical use case is the batch processing of ntuples. The root interface uses the shared libraries for SAM created in diskcache_i.

LIMIT ON TRANSPARENCY

The transparency of the data handling system's change in user analysis jobs has limits. SAM uses datasets to specify data the user requires whereas the DFC data handling system uses filesets, which are directories in which the files are copied and lists of files. The users therefore need to understand the concept of datasets.

In contrast to the old data handling system SAM delivers the files in the fastest possible order which is an advantage especially for distributed data handling because files which are already present at remote sites are analysed first. This however means that files are not delivered in the same (deterministic) order each time the job is executed.

CONCLUSION

In this article the CDF interface between user analysis jobs and the SAM data handling system has been described. The most important feature is the maximum of transparency of the data handling system's change to the user. The interface has been thoroughly tested and provides a reliable interface for users and therefore increases the user acceptance of SAM.

REFERENCES

[1] http://d0db-prd.fnal.gov/sam/

[2] Testing the CDF distributed computing framework, V. Bartsch et al., Prepared for Computing in High-Energy Physics (CHEP '04), Interlaken, Switzerland

[3] Automated recovery of data-intensive jobs in D0 and CDF using SAM, V. Bartsch et al., Prepared for Computing in High-Energy Physics (CHEP '06), Mumbai, India