September 8, 2008
CDF/DOC/COMP_UPG/PUBLIC/XXXX
Version 1.0

SAM-SRM Interface for Movement of MC Data

Manoj Kumar Jha
INFN-Bologna, Italy

Gabriele Compostella, Antonio Cuomo, Donatella Lucchesi, Simone Pagan
INFN-Padova, Italy

Doug Benjamin
Duke University, Durham, USA

Robert Illingworth
Fermilab, Batavia, IL, USA

Abstract

The CDF experiment has generated more than 3 fb^{-1} of raw data. As much as same amount of Monte Carlo(MC) data are needed for detector understanding and physics analysis. It is not feasible to produce these MC data on-site due to limitation on computing resources. The CDF remote sites or Grid Tier1 and Tier2 can be utilized for producing MC data. From our past experience, we learnt that one of the most important limitation in the heavy usage of off site resources is that the Worker Nodes(WN) were sitting idle for several hours just because another WN was transferring data to Storage Element(SE) at Fermilab. This situation leads to inefficient uses of computing resources and sometimes to the loss of the output from the worker node. Moreover, how to face the problem of sudden arrival of MC output data from different remote sites at same time to the SE at Fermilab. We found that this causes overloading of available file-servers and leads to failure of data handling part of MC production. Hence, a robust mechanism is needed for transportation of MC data from WN at remotes sites to SE at Fermilab.

Contents

1 Introduction

The CDF experiment has generated more than 3 fb^{-1} of raw data. As much as same amount of Monte Carlo(MC) data are needed for detector understanding and physics analysis. It is not feasible to produce these MC data on-site due to limitation on computing resources. The CDF remote sites or Grid Tier1 and Tier2 can be utilized for producing MC data. From our past experience, we learnt that one of the most important limitation in the heavy usage of off site resources is that the Worker Nodes(WN) were sitting idle for several hours just because another WN was transferring data to Storage Element(SE) at Fermilab. This situation leads to inefficient uses of computing resources and sometimes to the loss of the output from the worker node. Moreover, how to face the problem of sudden arrival of MC output data from different remote sites at same time to the SE at Fermilab. We found that this causes overloading of available file-servers and leads to failure of data handling part of MC production. Hence, a robust mechanism is needed for transportation of MC data from WN at remotes sites to SE at Fermilab.

The above inefficiency arises due to limitation on available bandwidth between WN and SE at destination sites. It can be overcome by first storing the MC output (from WN) temporarily in Storage Element (SE) closer to WN and, then transfer sequentially to storage element at destination site. Figure 1 illustrates the movement of data from WN to SE at destination site for different CDF Analysis Farms (CAF). Each headnode represents different CAF and there exists a SE closer to WN. The term *closer* means that the bandwidth between WN and SE is comparatively good or they are within the same Local Area Network (LAN). When the user's job end on WN, the output data is first temporarily stored in SE closer to WN and then moved it to SE at destination site. In this case, the waiting time of output data on WN has been considerably reduced due to the large bandwidth available between WN and its SE. After transferring output data from WN to SE, this WN can be assigned to other user's job. In this way the CPU resources available per unit time have been increased with respect to earlier case when the output data was being directly transfered from WN to SE at remote destination site.

Figure 3 represent the proposed model for transfer of MC data for a single CAF. In the rest of document, we will be talking about the model for a single CAF. The entities in the rectangular blocks constitutes different part of the model. Each SE acts as cache of a SAM station. We are using the features of SAM-SRM interface [2] for data movement, replication and caching. For the sake of completeness, a brief description on SAM and SRM has been presented in the Sections 2 and 3. A short introduction on SAM-SRM interface is being given in Section 4. Section 6 gives the detailed description of the proposed model.

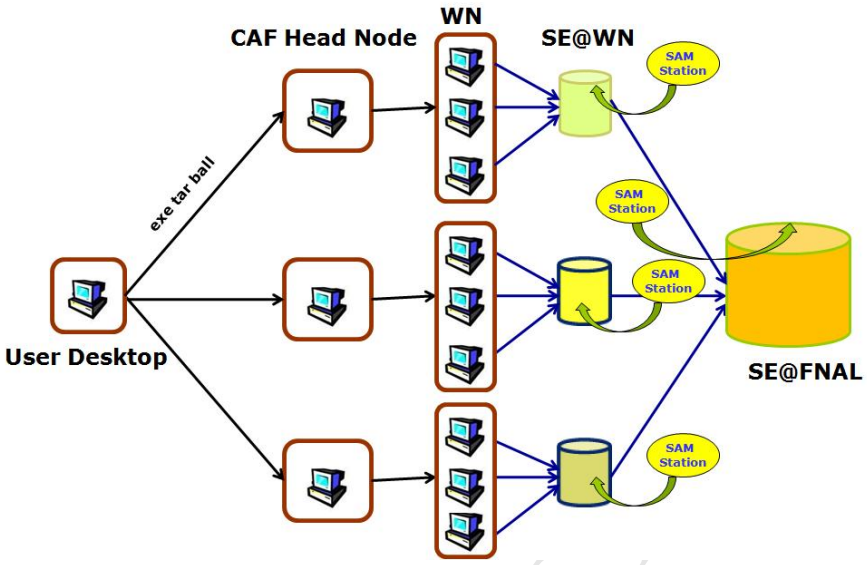

Figure 1: A prototype model for transfer of MC data from WN to SE at remote destination sites.

2 The SAM Data Handling System

Sam is a file based data management and access layer between the Storage Management System and the data processing layers. The goal of this SAM is to optimize the use of data storage and delivery resources, such as tape mounts, drive usage, and network bandwidth. In order to facilitate this goal, the primary objectives are:

- Provide reliable data storage, either directly from the detector or from data processing facilities around the world.
- Enable data distribution to and from all of the collaborating institutions.
- Thoroughly catalogue the data for content, provenance, status, location, processing history, user-defined datasets, etc.
- Manage the distributed resources to optimize their usage and, ultimately, the data throughput, enforcing, at the same time, the policies of the experiments.

The SAM system achieves these goals via a global network of cooperating software services, some of which are centralized in nature, like, for example the meta-data catalogue, while some are fully distributed. Among the distributed services, the principal and foremost is the SAM station, whose role is pooling together a set of resources at a site, such as storage, computing, and network systems, for the sake of data management.

The SAM system achieves data storage and distribution by coordinating the access of the globally distributed SAM stations to globally distributed storage systems. In

this regard, the primary and foremost service offered by the data handling system is data replication. The service is automatically triggered by clients unable to directly access the data, due to topological constraints. Using delegation among stations, SAM transfers the data to a storage system accessible by the client, cataloguing, at the same time, the intermediate data locations. The route decided for the transfer depends on the selection criteria of the initial replica and on the topology of the stations, parameters that can both be tuned to shape the network traffic. In order to achieve replication, SAM implicitly relies on its data movement service. For this service, data transfer is a layer of abstraction independent of the protocols, which are implemented, in the end, through a plug-in mechanism. In addition, to overcome possible errors during data transfer, SAM implements reliability in the form of data integrity checks, automatic retrial mechanisms.

To catalogue the data, SAM rely on a central, well maintained relational database, managed at Fermilab. The information stored in the database ranges from system configuration, such as available resources and users, to data description (raw detector data, derived data, binaries, etc.). A particular effort has been put into the design of a meta-data schema flexible enough to allow custom defined parameters. Table 1 shows the list of presently available attributes which are used for defining a meta-data corresponding to filesets. In this framework, users can define datasets giving logical names to queries in the meta-data parameter space. Another fundamental aspect of the schema is the data processing history, which, combined with the meta-data catalogue, provides a record of the data provenance, an aspect crucial for the reproducibility of the physics results. We will utilize the data processing history from SAM for book keeping and monitoring of MC jobs output data flow from WN to SE at destination site.

3 Storage Resource Manager(SRM)

There is a proliferation of custom storage solutions at high energy physics. SRM interface provides a standard uniform management interface to these heterogeneous storage systems, providing a common interface to data grids, abstracting the peculiarities of each particular Mass Storage System. Designers of higher level grid middleware can concentrate on function rather then compatibility with all systems involved, as illustrated in Figure 2. The SRM standard supports independent SRM implementations, allowing for a uniform access to heterogeneous storage elements. SRMs support protocol negotiation and reliable replication mechanism. SRMs allow site-specific policies at each location. Resource Reservations made through SRMs have limited lifetimes and allow for automatic collection of unused resources thus preventing clogging of storage systems with "orphan" files.

Attributes	Descriptions
applicationFamily	Valid application family that produced this file.
crc	Cyclic Redundancy Checksum(CRC) value.
dataTier	Data tier.
datastream	Physical datastream name.
endTime	End time of the physics run associated with this file.
eventCount	Count of the physics events contained in the file.
fileContentStatus	Status of the contents of the file, regardless of location.
fileFormat	Format of the file (must be a registered format value).
fileId	DB-assigned unique identifier.
fileName	Unique file name.
fileSize	File size.
fileType	NonPhysicsGeneric, physicsGeneric, importedDetector, derivedDetector, importedSimulated, derivedSimulated, or an experiment-specific known file type.
firstEvent	First physics event number contained in the file.
group	Physics work group responsible for this file.
lastEvent	Last physics event number contained in the file.
lumBlockRangeList	List of (lumblockMinId,lumblockMaxId) pairs.
params	Processing params associated with this file.
parents	List of the files from which this file was derived.
processId	Consumer process id of the analysis process that created this file.
runDescriptorList	List of runs (runType and runNumber) associated with this file.
sourceSplit	Split value for split/merge file lineage.
startTime	Start time of the physics run associated with this file.

Table 1: List of attributes available in SAM for definition of meta-data.

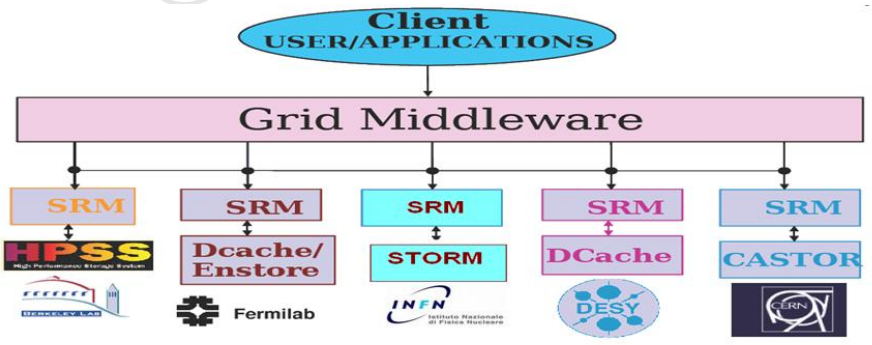

Figure 2: Storage Resource Manager as interface between SE and grid middleware.

4 SAM-SRM Interface

The SAM-SRM interface apply SAM data handling policies over the generic storage system managed by SRM protocol. The premise of the concept is similarity between SAM managed disk storage and SRM managed storage in terms of many supported operations. In particular such operations as transferring file, removing file, building directory listing, and retrieving file meta-data are common for POSIX file systems and are also supported by SRMs.

SAM disks are used as root locations to place data files. Disks also have size that limits disk location space usage by SAM. The Logical disks and SAM file replacement policy constitute SAM cache that brokers resources between storage elements and storage element clients. Naturally, the same concept is reused to point SAM to an SRM location where files should placed or removed from. More information on SAM to SRM mapping can be found in specification document [2]. SAM-SRM interface is expected to support following properties.

- User should be able to pick SRM location where he/she expect SAM to place and access data.

- User expects SAM to place data to a chosen location on demand and according to polices set by the SAM station configuration (pre-fetching, fair share, replication algorithms and etc.) and SRM. Data placement on demand assumes that user does not have explicit knowledge about data source, data status at the storage or overall topology of underlying storage deployment. SAM, therefore, is responsible to ensure that data is moved and made accessible to a user in a transparent manner.

- User expects SAM to translate logical location into one of the following strings that should comply with access protocols: dcap , gsiftp. These strings should be made available to application such that latter can access actual data bits. The exact choice of the protocol is either static or dynamic.

- User expects SAM to register data in the replica catalog in order to reflect successful placement attempts. Registered locations should provide enough information to build a successful query to SRM by registering or external stations.

- User expects SAM to synchronize on demand replica locations with actual status of data in SRM storage.

- A minimal policy SAM should support is the policy of replacement based on the storage size and storage type (permanent/durable).

5 CDF Analysis Framework

The CDF Analysis Framework(CAF) is a large network of computing farms dispersed over a large network of resources designed to allow users to run multiple instances of the same CPU-intensive jobs. The heart of the CAF resides at Fermilab. However, there are several other CAFs around the world(DCAF), including, but not limited to, SDSCCAF in San Diego, MITCAF at MIT, CNAF at Bologna and many other facilities. Submission, monitoring,and output are authenticated by user's FNAL.GOV kerberos ticket, allowing users to access these facilities from any kerberized machine in the world. The CAF is based on the Condor system. In order to manage security more effectively, the CAF has a single node that centralizes all of the user's command and control functions. Following are the three types of computers implemented in the CAF system:

- Headnodes
- Monitoring Nodes
- Worker Nodes

Headnodes are the main access point to the CAF system. It is through this node user submit their job and get any job-related notification. Monitoring Nodes are the web servers that hold the CAF's web monitoring code. Web monitoring allows user to monitor the location and the status of his/her jobs via the web. WNs are where all user's work gets done. Every WN in the CAF is divided into several Virtual Machines(VMs), each of which is capable of running an independent job.

The CAF framework sends user job to the CAF headnode and in return user's gets a Job ID(JID). Using the JID, user monitors his/her job remotely. After that a normal job enters the queue at this point, starting what is referred to as a dagman, sort of a central manager that runs user job. The dagman starts entering sections into the main queue, where they are given a status(idle, wait or running). After the job completes, the job gets return to the headnode. The headnode arranges to send an email to user's email address reporting the status of the job. For each job segment, the CAF system software tar up and gzip user's working directory and send it to the specified output location.

6 Proposed Model

The proposed model relies on integration of SAM with SRM. We used SAM because it is the default data handling framework of the CDF. SAM offers tools to describe storage deployments as well as implements policies of data movement, replication and caching. The reasons for using SRM are to avoid unnecessary complications which may arise from different SE at different grid access sites. SAM-managed storage elements are organized as configured station consumption sites with limited by size on storage

system. SE and SAM file replacement policy constitute SAM cache that brokers resources between storage elements and storage element clients. To efficiently manage data flow, SAM cache is operated with pre-set assumptions on costs of data access for its storage elements. In particular, cost of access is assumed to be uniform across the SAM cache. Moreover, the data can neither appear nor disappear without explicit request from SAM. Data management and its replications are the inherent features of the proposed model.

Figure 3 shows a prototype model for movement of MC data from WNs to the SE at the remote destination site. The destination SE in our case is at Fermilab. It can be any of the remote DCAF so that data produced at other DCAF can be transfer there. Following are the components of the proposed model.

- A SAM station and temporary space of around few Tera Bytes(TB) on SE which is closer to WN.
- Another SAM station and SE at the destination site. Since, the data on SE are managed by SRMs, the physical location of SAM stations don't matter in this model.
- As output location of MC data, users would have to indicate the name of nearest SAM station available to them.

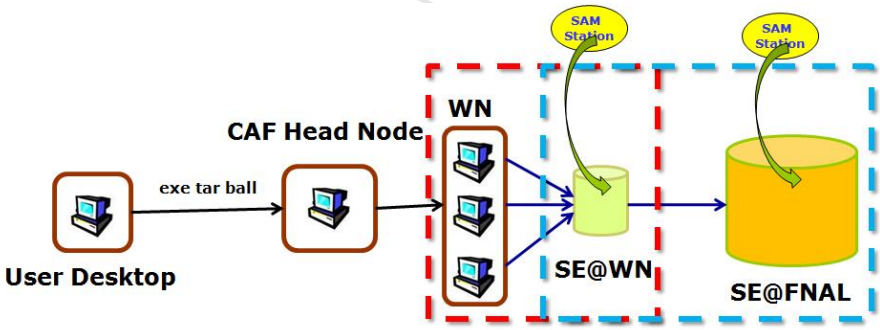

Figure 3: Proposed model for transfer of MC data for a single CAF.

Following are the steps involved in the proposed model for output data transfer from WNs to the SE at remote destination site.

- User jobs run on the WNs. When a segment of job finishes on the WN, the CAF framework will tar up and gzip the user's working directory and generate its meta-data. The meta-data is a file of the order of few bytes which contain all the information related to MC output file. Table 2 shows the minimum list of attributes which will be available in the generated meta-data. The attributes which are marked by * are not available in the present version of SAM as mentioned in Table 1. The criteria for defining the list of attributes in meta-data is based on:

  - Ability to register the output file using its meta-data in the SAM database.
  - Some of the attributes of meta-data will be used for retrieving and management of MC file from the station cache. For example, the user would like to get his/her file corresponding to particular section of a JID, or he/she would like to get the files generated on a particular farm on a specific day, and so on.

- We assume that there exists folder like "fromUsers", "upload" and "failed" with respect to allocated location on SE closer to WN.

- CAF framework will create a folder "fromUsers/JID" on SE corresponding to each JID submitted by the user.

- CAF framework will transfer gzip tar output and its meta-data from all the segments of a JID in the "fromUsers/JID" folder of SE. Tools like srmcp [7] or gLite FTS can be used for transfer of files from WN to SE. We found that srmcp tool exists on all the OSG sites. We would have to confirm about the availability of srmcp on other sites.

- Following are the activities initiated by a process which will run on the PC attached to SAM station.

  1. Exit if the previous session is still running.
     - Process ID(PID) file exists and the process is still running.
  2. Exit if SAM is down.
     - "sam dump fss" works or not.
  3. Save the PID to a file.
  4. Checks for availability of new folder in "fromUsers". The new folder name corresponds to the user's JID. .
  5. If the number of file is twice the number of sections(which can be get from the meta-data), then it moves the "fromUsers/JID" folder to the "upload/JID" folder.
  6. Build a "to_submit" list from "upload", "fromUsers" and "failed" folders.
  7. Remove the files which are still in the SRM_STAGER queue from the "to_submit" folder list.
  8. Creates a dataset using the meta-data available in the submitted list and upload it to cache of the SAM station closer to WN using the command

     sam upload −−destinationPath=<destination location>
     −−sourceFile=<fully-specified path to the file>

     If the process fails in uploading data to the cache of SAM station, the folder "fromUsers/JID" will be moved to "failed" folder. This step is under

discussion. At present, it is difficult to say whether each segment of a JID will be registered into SAM database. Other option is to use previously registered dataset corresponding to a user id and then add meta-data of all segments corresponding to a JID. Similarly, for other JID of the same user. These things depend on schema of meta-data.

9. The status of the command in step 8 needs to be informed to user. It can be done either sending mail from headnode or from the process itself inform the user. If the process inform the user, then we need the email id of user as list of attributes in the meta-data.
10. Update the heart beat and monitoring file.
11. Copy the dataset (generated in step 8) into the destination station (output location given by user) using the command

    sam get dataset −−definitionName=<NameOfDataset> −−station=<destinationStation>
12. Add location of dataset corresponding to cache of destination station

    sam add location −−fileName=<value> −−replicaLocation=<value>
13. Delete location of dataset from the cache of station closer to WN

    sam erase location −−fileName=<value> −−replicaLocation=<value>
14. Update the heart beat and monitoring file.
15. Remove the “upload/JID” folder created in step 5. In this way, the used space on SE closer to WN will be used again.
16. Folders available in ”failed” folder which are not in the SRM_STAGER queue, move the ”failed/failedJID” to ”upload/JID”.
17. Remove the PID file.
18. In the end, the process skips to step 1.

The validation team will validate these new files and correct files will be moved to Fermi enstore robot for storage using the command

sam store −−resubmit −−descriptionFile=<meta-data description file>.

The above step needs creation of good list of files from the station cache. Accordingly, meta-data will be created for each set of good files and then it will be added in the existing dataset for storage in tape using the above command. More detail procedure will be given in coming days.

Attributes	Description
date*	Date on which file gets created
email*	Email id of the user
farm*	Name of the CAF/DCAF
fileName	Unique file name.
fileSize	File Size
nSections*	Number of sections submitted for a job.
processID	Process ID of the user's job
stationName*	Output location
userID*	User's name or ID

Table 2: Minimum list of attributes needed in the meta-data for proper functioning of the proposed model. Attributes which are marked by * are not available in the present version of SAM.

7 Job Submission

User will still use the same interface "CafSubmit" for submitting job on CAF/DCAFs. User would have to provide the name of SAM station in the option "−−outLocation" where he/she wants the job output. The job output reside in cache of the station and user would not have to worry regarding its physical location.

8 Job Output

Since files reside in station cache. A file can be accessed by means of creating a dataset through list of attributes. Suppose a user wants all the file corresponding to a JID(say 123456) on his/her desktop. Then the user would have to do the flowing steps

- Define the dataset and take a snapshot:
  sam create dataset definition −−defname=<dataset_defname> −−group=test −−dim="processID 123456"
  sam take snapshot −−defname=<dataset_defname> −−group=test

- Copy file on the user's desktop
  sam get dataset −−defName=<dataset_name> −−group=test −−downloadPlugin=<file transport tool>
  If the station is at FNAL, user can use "downloadPlugin" as "dccp". For uniformity, we will try to support "srmcp [7]" as "downloadPlugin" for all the stations.

The above steps can be integrated into the existing "icaf" tools for getting the file from station cache to user's specified location. It will make the life of user's easier !

It should be noted that file will reside in cache as long as it don't gets fill. After that older file starts getting deleted so that new file can be store in the cache of station.

The file deletion policy will be similar to that of file deletion in SAM. For a file to have longer life time, the space of cache on station should be sufficiently large in comparison to the filling rate of station cache.

9 Future Plan

There are also some other spin off from this study. One can use this model for transporting real data from productions site to the WN at remote site for further processing of real data.

References

[1] CDF Homepage, **http://www-cdf.fnal.gov**

[2] SAM-SRM design document, **https://plone3.fnal.gov/SAMGrid/Wiki/SAM-SRM-Design.doc/download**

[3] SRM Working Group Homepage, **http://sdm.lbl.gov/srm-wg/**

[4] SAM Homepage, **http://d0ora1.fnal.gov/sam/**

[5] CAF Homepage, **http://cdfcaf.fnal.gov**

[6] dCache Homepage, **http://www.dcache.org/**

[7] srmcp Homepage, **https://srm.fnal.gov/twiki/bin/view/SrmProject/SrmcpClient**

[8] Condor Homepage, **http://www.cs.wisc.edu/condor/**