SAM-SRM integration specification and design

A. Baranovski, R. Mathur

Concept

The base concept of the integration effort is to 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 metadata are common for POSIX file systems and are also supported by SRMs.

SAM storage resource presentation

SAM-managed storage elements are organized as configured station consumption sites with limited by size logical disks. SAM disk is used as a root location to place data files. Logical disks and SAM file replacement policy constitute SAM cache that brokers resources between storage elements and storage element clients. T o 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 not disappear without explicit request from SAM.

Assumptions on SRM interface and its implementation

SRM storage is expected to support following properties:

1) Get file. Translates logical file path into transfer url (location access cookie) compatible with requestor transport.
2) Put file. Allocates space and translates file path into location access cookie compatible with requestor transport.
3) Release file. Deallocates resources associated with calls 1) and 2)
4) Copy file. Duplicate data file between 2 storage elements
5) Must be able to create a file path on the storage.
6) Must be able to free storage space by removing a file.
7) May limit number of concurrent accesses by number that can be less than total number of requests in the system.
8) Must allow data read access by credentials not originally used to copy, put, or get a file. An important requirement meant to coordinate and ensure shared storage management.
9) Must allow querying for file and directory metadata.

SRM to SAM resource mapping

Despite been similar in many basic file system manipulation operations, there are still policy level differences between SRM and POSIX storage that need to be addressed in order to transparently integrate SRM storage in SAM. These differences can be summarized below:

1) In order to access a file, SAM managed disks do not require explicit resource allocation outside of the program context. The resources are allocated on a POSIX level after a file has been opened for actual reading/writing. In contrast, SRM requires every file access attempt preempted with explicit “get” or “put”.
2) On certain types of SRM managed storages a file may implicitly (i.e.

without explicit action from the storage user) “disappear” or change its “cost of access” category.

3) SRM requires explicit release op invoked on a resource handle irrespective of the state of the actual read or write operation.
4) In contrast to uniform cost of access common for the POSIX file systems , SRMs differentiate the cost of data access on a per file basis.
5) In SRM, data resource allocation is not managed within the same API and parameter space as actual data read and data write.

Each listed policy need changes in SAM beyond the implementation of trivial mapping of the copy, put, rm, ls, mkdir commands. In particular, SAM caching subsystem will need to be re-factored to address issues specific to loose binding between SAM station and physical resource in a way that new approach is uniform and remains within existing SAM paradigm.

The first step of the approach is to reuse SAM “cache” disks as the concept to describe SRM resource to SAM. SAM disks are the objects that contain memory of files requested by users in order for the internal cache management policies to make decisions on data placement. The same way, disks will be used to keep memory of files requested by SAM via SRM. Reuse of the existing SAM interface to describe and manage SRM resource will allow imposing uniform data management policies over durable, permanent and volatile storages.

Second, in the new model SAM will need to address loose binding between storage and SAM by adopting a mechanism that handles uncertain state of the files in SRM. This implies a change in error handling/recovery policies that in turn, may call for special optimization techniques to improve efficiency.

Third, SAM will be responsible for managing user load of the SRMs. This entitles allocation/deallocation of the SRM handlers, keeping track of timeout conditions and actual logical to transfer url conversion.

Types of storage

Below is list of supported SRM storage elements and suggested functionality changes in SAM needed to address specific aspects of respective storage type.

Volatile storage

Volatile storage may displace its own files voluntarily unless a file is explicitly used. Therefore, SAM cache should expect that data delivery/pre-staging history may not be entirely accurate. In order to ensure that replica catalog and storage state are consistent SAM must check data availability each time file request is made. Such synchronization checks should be done in a “lazy” fashion and should result in cache state corrective actions issued against the program. Consecutively, the srm locations will have to be unregistered from SAM if corresponding data files have been found physically missing.

Volatile storage is of a finite size and may have implicit space cleanup policies. While cleanup policies can not be overridden, the explicit file erasure requests from SAM will still be required to only serve advisory function.

Durable storage

Durable storage stores and removes file only by explicit request of the client. Consecutively, SAM has to explicitly manage the SRM space by issuing erase requests in order to keep SRM file quota under the configured limit. In that case, the srm location of displaced files will need to be unregistered.

Permanent storage

Permanent storage never displaces files. However, it is typical that data access cost of the permanently stored files may change in accordance to storage internal policies. Minimizing data access cost is an important aspect of the experiment data handling. Which is why, in order to improve the efficiency of the SAM/SRM model, the cost factor will be explicitly managed.

Explicit cost factor management will rely on “active” and “passive” request prioritization

“Active” request prioritization will use SRM to learn of access costs for specific “lookahead” fraction of files.

“Passive” request prioritization will rely on SAM feature to first dispatch data that reside in SAM cache. The use of the SAM cache to dispatch SRM requests may help to optimize number of prestage/”cost inquire” requests to SRM in a scenario where SAM cache is modeling cost of access function of the underlying SRM storage. Thus, the mechanism may contribute to better cache hit rates while reducing SRM overhead. The question of the impact of the the model on actual SRM cache usage efficiency depends on such configurable factors as: size of the SAM cache, SAM file replication policy. Both parameters may require tuning specific to the storage at hand.

For this type of the storage, the srm location must remain registered in the DB reflecting permanency property of the SRM.

Transfer URL proxy-ing and management.

Each cached file may have "srm request id" and the list of transfer (access) urls attached to it as created either by invoking pre-stage or copy commands. The transfer urls generated by “get” command cannot be shared between multiple projects and therefore are used immediately after the file was pre-staged. The transfer url generated by invoking “copy” commands may not be re-used due to the conflict between command issuer (SAM) and the transfer url consumer (analysis job) principals.

Because historically the issues of synchronization between replica catalog and storage content can only be addressed by the SAM station, the translation of logical urls to transfers urls (the place where replica catalog/SRM state divergence can be typically detected) may not be reliable delegated to the client. For the same reason, SAM must request translation of the logical urls to transfer urls each time file is about to be dispatched to the client.

SRM Transfer management

Assuming that all files have SRM locations, station assigns destination of the SRM transfer taking into account existing consumption map and request priorities. Cache fit algorithm, routing and regular delivery constraints are used to select transfer destination for any given request. As a result, cache fitting algorithm generates list of final transfer requests for the SRM stager.

The cache fitting algorithm is driven by individual project's limits on number of concurrent requests with preconfigured “lookahead” number. The resulting aggregated list of transfer can be set large enough to benefit from SRM internal logic to reorganize deliveries (including those requests that target SRM’s internally cached files) in a more optimal way. In addition to that, station will implement a provision to re-sort the list according to priorities generated by the estimator service. This provision works up to active request priority management requirement as describe above.

Station interprets transfer request to either invoke “get” or “copy” operations: The exact decision is based on analysis of the request source location. In a detail, If source location of the request it determined to be one of the station's own consumption disks - srmGet is used, otherwise srmCopy is used. Regardless of the method, resulting SRM request must reference valid SRM location (whether new or existing if srmGet was used), transfer urls and request id. The outcome of the SRM call is then cabled to station, project, and finally to the analysis job.

Location mapping

Mapping the location of the file presents us with the problem of directory structure at the destination. The current SAM implementation simply copies all files into a base directory in the station cache (as configured by disk’s root location). This does and will continue to cause problems with respect to efficiency and maintenance.

An alternate solution is to append to the "base directory", the entire path from which the file is being copied. This, while it solves the earlier problem, causes a new problem of always-increasing path lengths, and will result in an unwieldy directory structure.

A possible solution is to use the "replica locations" feature from SAM and choose the most suitable directory structure from the list of locations. A simple solution would be to use the directory path that is the shortest of all the replica locations. This ensures that the particular file will always be stored in that particular subdirectory structure.

Another possible solution would be to use the path that is similar to the current protocol i.e. in the case of an srm:// transfer, use the directory structure from the replica location that is also an srm:// location.

The namespace replication policy may thus help to implement common requirements on the maximum number of files per location across all supported storage elements.

Summary of the changes to common SAM services

Estimator

Estimator is a java based service that will assign priorities to transfer requests based on file’s cached status at the transfer source. The priorities of the requests that source from the cached files are valued highest and lowest otherwise. The neutral priority is chosen for requests for which source file cached status can not be determined or supported.

The numeric priorities will weight in the order in which station cache fit algorithm considers data placement. In particular, higher priority requests will be executed first.

The “Estimator” service will act on behalf of the “sam” principal

Stager

Stager is a java based service that adapts SAM callback based communication model to poll based SRM communication model for “copy”, “ls”,”mkdir”,

“get”,”delete”,”release” ops.

Stager will support srm request ids as well notions of logical and transfer urls.

Stager may act on behalf of different user principals. For the non “sam” principal, stager may use MyProxy secure service to acquire delegated credentials to execute requested operation. Delegated credentials must have limited expiration time as specified by the requestor.

Project

Project will communicate with station and user analysis data file names, storage names, transfer urls and events that will regulate the pace of the data flow. Project will maintain the user’s request context to increase tolerance of the system towards station failures.

Project context will include srm request ids and will de-allocate request ids before SAM station is notified of change in state. The notification sequence will ensure that station does not dispatch new requests before used resources are properly released. That is to implement consistency between SAM data handling and SRM storage states.

Station

Station will use estimator service to actively prioritize requests. The active prioritization will be based on number of outstanding requests weightened by the “lookahead” rule.

Station will use stager with “sam” principal to copy files that do not have srm location registered in SAM. Station will use user principal to “get” or pre-stage files that reside in SRM and have their location registered in SAM.

Station will list and translate logical to transfer url each time a file is about to be opened by the analysis process.

Station will supply file name, srm request id, logical and transfer urls to the project.

Station will delete files from volatile and durable SRMs as they leave SAM cache and will consecutively un-register their SRM locations from SAM.

Station will implement location mapping rules to support experiment namespace convention across all supported storage elements.

Security model

The model of the integration assumes that data is shared between storage users such that common data handling can be enacted on behalf of the experiment group. Achieving this function means overcoming security and permission restrictions normally used to protect storage service in multi user environment. Hence, as a compromise between security and manageability SAM will act on behalf of the unique principal to stage and remove experiment data. At the same time SAM will proxy actual access requests using user's own principals. Proxying user credentials will ensure that SRM correctly prepares transfer urls in agreement with credentials that are normally presented by the analysis job at a time data is read.

The exact schema to proxy user credentials has not been fully defined. Tentatively, SAM stagers can use MyProxy to delegate user's credentials to SRM.

Db Schema consideration

Station employs existing DB schema to store SRM locations. Station uses "srm://" prefix to distinguish these locations from existing ones.

Station will dynamically create new SRM location explicitly when files are cached at SRM consumption site unless a file has already been registered. Station will un-register the location if all attempts to import a file fail.

Station components data and basic execution flow.

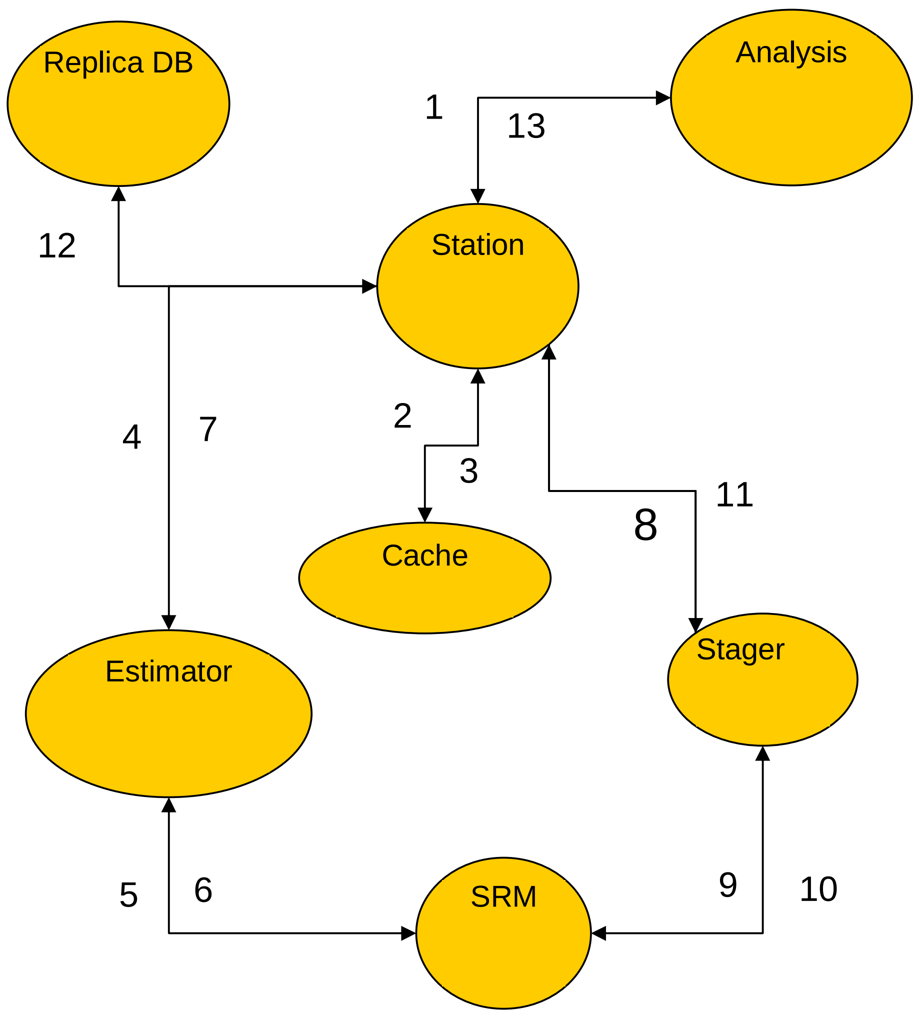
1) start project
2) Is in cache?
3) Yes/no . If yes go to 8 , otherwise go to 4
4) get priority
5) srm get metadata
6) return metadata
7) return priority
8) request transfer / get
9) srm get/copy

10) srm get/copy done

11) transfer request complete

12) register new SRM replica

13) file is available


Replica DB
Analysis
1
13
Station
12
4
7
2
3
Cache
11
8
Estimator
Stager
5
6
SRM
9
10