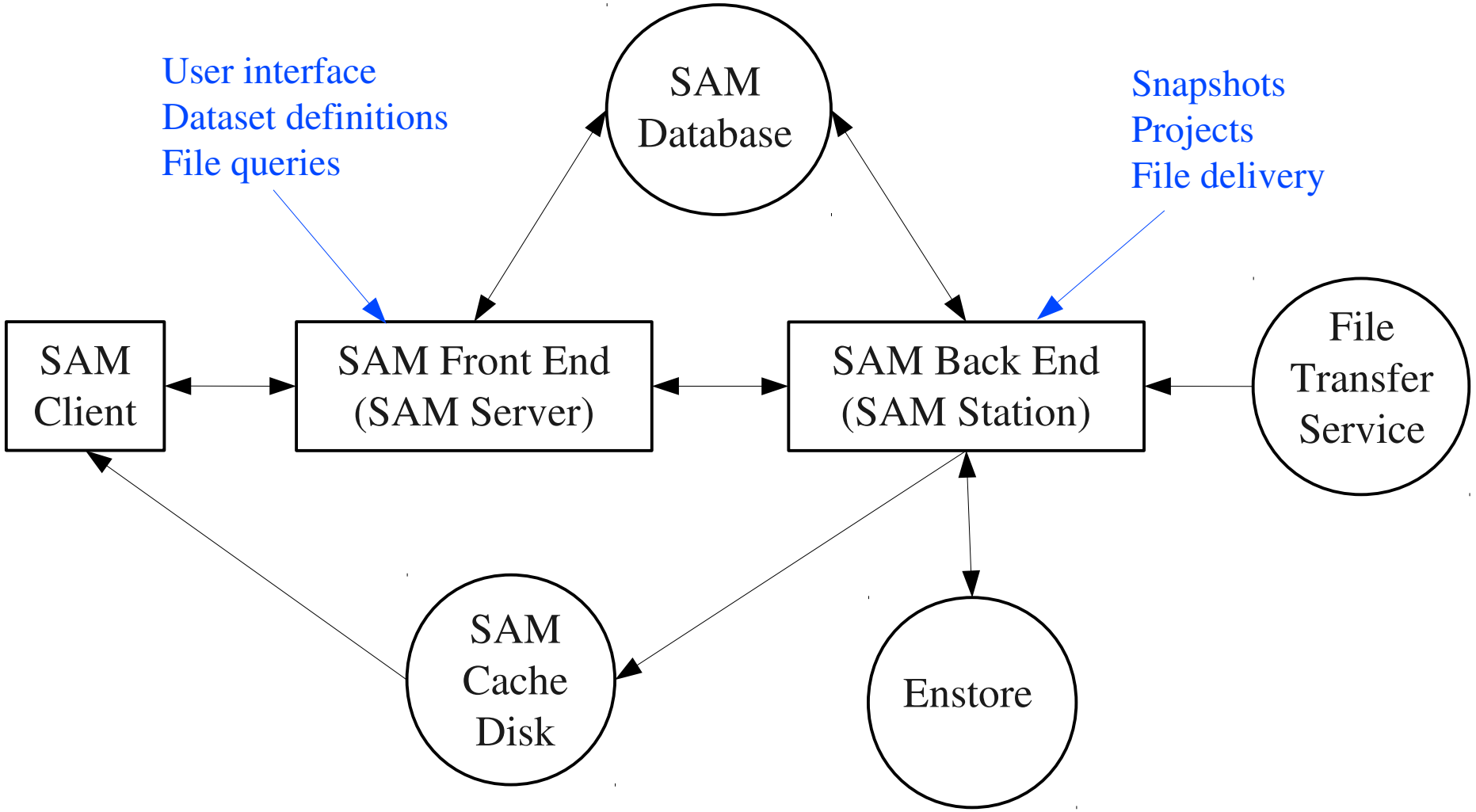
Art and Sam Integration

General Larsoft Meeting
May 8, 2013

H. Greenlee

Data Handling Overview



Sam Clients

- All sam clients have in common that they send requests to the samweb http server (http://samweb.fnal.gov:8480/sam/uboone/api)
- Samweb (setup sam_web_client).
  - Line mode client (samweb -e uboone <subcommand> …).
  - Python client (import samweb_cli).
- Ifdh client tools (setup ifdhc).
  - Line mode client (ifdh <subcommand>)
  - C++ client (class ifdh, not art-specific).
- Art client (setup ifdh_art).
  - Wraps ifdhc c++ sam client as art service (IFDH service), and provides sam-capable instances of file delivery and file transfer services.

Art Sam Use Cases

- Generating sam metadata for output files.
  - This use case does not actually require sending requests to the sam server. You just have to know what metadata you want to associate with each output file.
    - Not sure if there is a way to extract sam metadata from input files.
- Fetching files from sam.
  - This use case requires that the art program send “fetch next file” type requests to the sam server.
  - Sam project has to have been started on the sam station (probably external to art program).

Sam Metadata

- If you have access to microboone docdb, refer to document 2414 for a description of microboone sam metadata.
- Sam metadata fields can be classified as either:
  - Per-file or per-job
  - Standard (built-in) or experiment-specific.

Sam Metadata (cont.)

	Per-Job	Per-File
Standard	File Type Application Family Application Name Application Version Group	Data Tier Data Stream File Format Parent Files Event Count First and Last Event Run Numbers and Types
Experiment-Specific	Fcl Name Fcl Version Project Name Project Stage Project Version (more)	Filter Name (more)

Fields shown in red are implemented in the current version of art

Generating Per-Job Metadata

- Art supplies a service called FileCatalogMetadata for generating standard per-job metadata (currently file type and application family, name, version).
- In the last week, I added a service called FileCatalogMetadataExtras in Utilities package for generating arbitrary experiment-specific per-job metadata.

Example Job File

- FileCatalogMetadata is configured in “services” section of job file.
- FileCatalogMetadataExtras is configured in “services.user” section.
  - Experiment-specific metadata specified as arbitrary number of name-value pairs.

```
services:
{
  FileCatalogMetadata:   @local::art_file_catalog_mc
}
services.user.FileCatalogMetadataExtras:          @local::microboone_file_catalog
services.user.FileCatalogMetadataExtras.Metadata: [ "group",      "uboone",
                                                    "fileFormat", "root",
                                                    "fclName",    "standard_reco_uboone.fcl",
                                                    "fclVersion", "development" ]
```

Generating Per-File Metadata

- Standard per-file metadata is generated in art module RootOutput.
  - Currently generates only data tier and and data stream fields (but needs to generate much more, including dynamic fields, like parents and runs).
- My currently favored approach (not the only approach) for generating experiment-specific per-file metadata is to derive an output module from RootOutput.
  - Yes, a module can inherit from another module, provided the base class module has a header file (RootOutput does).
  - RootOutput should provide a virtual method for adding experiment-specific metadata.
    - RootOutput kind of does this already, except relevant methods are private.

Generating Unique Output Filenames

- Sam has a requirement that all filenames be unique.
  - Sam does not have a concept of hierarchical directories or paths.
  - We did not generate unique file names for microboone 2012 MC challenge files.
- To this end, I have proposed adding the feature in RootOutput to generate output file names according to a template that can include the following items.
  - (Parts of) input file name.
  - Time stamp.
  - Environment variables (e.g. $CLUSTER, $PROCESS).
  - File sequence number.

Summary of Proposed RootOutput Changes

- Generate all standard metadata.
- Add a hook for adding experiment-specific metadata.
- Add ability to generate unique output file names according to a template.

Storing Sam Metadata in Art Files

- FileCatalogMetadata and RootOutput store sam metadata for each file in an internal sqlite database.
  - Data is stored as string name-value pairs.
- Standard art program “sam_metadata_dumper” can be used to generate a human-readable dump of internal metadata.
- Each experiment will have to provide a “metadata extractor” script to produce properly formatted (.json) metadata files for use by sam utilities.
  - Metadata extractor should add fields that aren’t generated internally.
    - File name.
    - File size.
    - File checksum (crc).

Sam_metadata_dumper Example

```
% sam_metadata_dumper standard_reco_uboone.root

File catalog metadata from file standard_reco_uboone.root:

0: applicationFamily  art
0: applicationVersion development
0: fileType           mc
0: process_name       Reco
0: group              uboone
0: fileFormat         root
0: fclName            standard_reco_uboone.fcl
0: fclVersion         development
0: dataTier           reconstructed
0: streamName
0: runType            physics
0: run                1
0: subRun             0
0: firstEvent         1
0: lastEvent          3
0: eventCount         3
0: startTime          1367960178
0: endTime            1367960256
0: parent             single_gen_uboone.root
--------------------------------
```

Fetching Files from Sam Using RootInput

- RootInput interacts with two kinds of abstract services, called a "file delivery service" and "file transfer service."
  - File delivery service has an interface that returns the URI of the "next" next file.
  - File transfer service fetches a file pointed to by a URI and returns a path on the local scratch disk.
- Art supplies two concrete implementations of these services, called TrivialFileDelivery and TrivialFileTransfer, which know how to handle local files and lists.
- Ifdh_art supplies sam-capable versions of these services, called IFCatalogInterface and IFFileTransfer.
  - Can be configured in user's job file, e.g., to fetch files from a Sam project.

Managing Sam Projects

- In principle it is possible to start a sam project in an art program (e.g. using IFDH service).
- However, in most cases (including projects with more than worker), it will necessary to start projects external to the art program.
  - Starting sam projects will have to be integrated into the batch submission system somehow.
  - I believe (haven't verified) that jobsub_tools provides some support for this (but will also require support from experiment-specific batch submission scripts).