The sam_upload: a tool to store and to catalogue files with SAM

Donatella Lucchesi
on behalf of the CDF SAM folks and the "skimmers"

Credits for the infrastructure I refer to go to SAM team and in particular to Fedor Ratnikov and Armando Fella

Outline:

- DFC approach
- SAM approach
- An easy way to store files: sam_upload
- Examples

Store Data Files

$$\begin{tikzcd}
 & \text{Store file = “write metadata” + “write file”} \arrow[dl] \arrow[dr] & \\
\text{Note somewhere file information} \arrow[d] & & \text{Write physically file on disk or tape} \\
\text{A napkin, your logbook, a web page or a database} & &
\end{tikzcd}$$

DFC Approach

File metadata:

-File name (key)
- Size
- Number of events
- First run/event
- Last run/event
- Contributing runsections
- Fileset
- Dataset
- User/date info
- Online and Offline luminosity

- Files grouped in filesets and moved to Enstore
- Group of filesets $\rightarrow$ dataset
- CDF dataset defined before the file is stored
- Metadata are stored by:
  - AC++ OutputModule
  - DFCFileTool

It works and easy to use

Store only data files (now also stntuple?)

SAM Approach

- Files may be of different well defined types:
  data, MC, ntuple, etc.
- Each file type has a minimal set of required metadata
- A LOT of other metadata types could be defined
  and declared for the file
- No fileset concept
- "SAM dataset" = "metadata selection"
- SAM dataset associated with CDF dataset
  now dataset must be defined before any file is stored

SAM Approach: what is needed

To store a file with SAM we need to know:

- File to be stored *
- Metadata for the file
- SAM station responsible for data transfer
- Host where SAM stager process is running and where it is able to pick up the file *
- Final destination for the file store (tape or disk)
- SAM group to be associated with the file **

* In all actual CDF configurations SAM stager runs on SAM station only. Therefore file to be stored must be visible from the station and “host” is always the SAM station itself
** group “cdf” is suggested for some commands “test” is needed

The sam_upload: the precursor

Fedor Ratnikov: samStoreCdfFile

- ✓ Generates necessary set of metadata:
  - o Data and MC: extract from data
  - o Ntuples: generic metadata only (for the moment)
- ✓ Obtains Enstore destination using CDF autodestination server
- ✓ Requests SAM to declare and/or store the file
- ✓ Can work in 2 steps: declare only and store only file
- ➢ File has to be "visible" from a SAM station $\Rightarrow$ disks shared or users in .k5login of the SAM station

The sam_upload: the tool

All credits to Armando Fella who developed and maintains it

Strategy:

- Extract metadata using Fedor's implementation and declare file
- Use kx509 to convert kerberos ticket to user's GRID certificate (no disks share SAM station-CAF or other places or users in SAM station .k5login)
- Copy the file from anywhere to a SAM station (any depending where you want to store) using GridFTP
- Store the file to disk or tape

Sam_upload example: store files from desktop

You need:
- access to CDF code
- know in which dataset you want to write the file
- dataset must be a valid cdf dataset booked in advance

```
setup cdfsoft2 5.3.4 (or grater)
setup sam v7_1_10 –q infn_prd
setup sam_upload v2_0_13
sam_upload uploadOnTape --rename /
--dataset=dataset_name --host=fcdfdata064.fnal.gov /
--station=cdf-samstore file_name
```

station=sam station used to perform the transfer
(the default for CDF will be cdf-samstore)
host=name of the node hosting that sam station.
If everything goes fine you get a message before having back the prompt which tell you all the step done by the program.
The status has to be 0.

Sam_upload example: store files from CAF

The sam_upload command is the same.
Here a script example used in the B hadronic dataset skim.

```
#!/bin/sh -f
source ~cdfsoft/cdf2.shrc
setup cdfsoft2 5.3.4
#SAM setup and access dedicated db server
setup diskcache_i -q KCC_4_0 v2_06_15
setup sam v7_1_10
setup sam_upload v2_0_13
export CDF_USER_NAME=lucchesi
# Set Output Filename
export OUT_BSDSPI_PHIPI=BsDspi_phipi_${name}.out
# Set output dataset
export DT01=skit01
# Run your program
./DFinderExample.exe main_SKIM.tcl >& DFinder_$1.log
RETC=$?
```

Sam_upload example: store files from CAF

```
if [ $RETC == 0 ]
then
# Now upload the file on tape
  fileName=`basename ${OUT_BSDSPI_PHIPI}`
  sam_upload uploadOnTape --rename --dataset=${DT01} \
   --parents=${parentList} \
   --host=fcdfdata064.fnal.gov --station=cdf-samstore $PWD/${fileName}
   ret1=$?
   if [ $ret1 = 0 ]
   then
     echo " echo " Hadronic Skim storing succeeded and file deleted"
   else
     echo " Hadronic Skim: sam_upload failed $ret1 "
   fi
else
    echo "Hadronic Skim: DFinder failed with return code = " ${RETC}
   exit ${RETC}
fi
```

Sam_upload commands <options>

sam_upload	uploadOnTape	write to tape
	uploadDurable	write to disk
	cancelFileTransfer	cancel file storage request
	getFileStatus	monitor status of uploaded file

uploadDurable same options

```
sam_upload uploadOnTape \
--dataset=<dataset>          CDF dataset assigned to file (mandatory)
--station=<SAM station>      SAM station name  (mandatory)
--host=<SAM host>            SAM station  hostname (mandatory)
--generic                    to store data with minimal metadata
--analysisGroup              CDF analysis group (book)
--rename                     rename file according CDF convention
--parents=<prnt1,prnt2...>   list of parents declared for these files
--grabber=<command>          to extract metadata from any file.
--mc                         data are MC, not detector
<file names>
```

Sam_upload: options and commands cont'd

sam_upload	uploadOnTape	write to tape
	uploadDurable	write to disk
	cancelFileTransfer	cancel file storage request
	getFileStatus	monitor status of uploaded file

sam_upload cancelFileTransfer \

-- station=<SAM station>	SAM station name (mandatory)
-- filename=filename	SAM station hostname (mandatory)

Sam_upload: options and commands cont'd

sam_upload	uploadOnTape	write to tape
	uploadDurable	write to disk
	cancelFileTransfer	cancel file storage request
	getFileStatus	monitor status of uploaded file

```
sam_upload getFileStatus \
--station=<SAM station>      SAM station name  (mandatory)
--host=<SAM host name>       SAM station  hostname (mandatory)
<file names>
```

User WEB page under construction, for the moment send email to armando.fella@pi.infn.it for more information

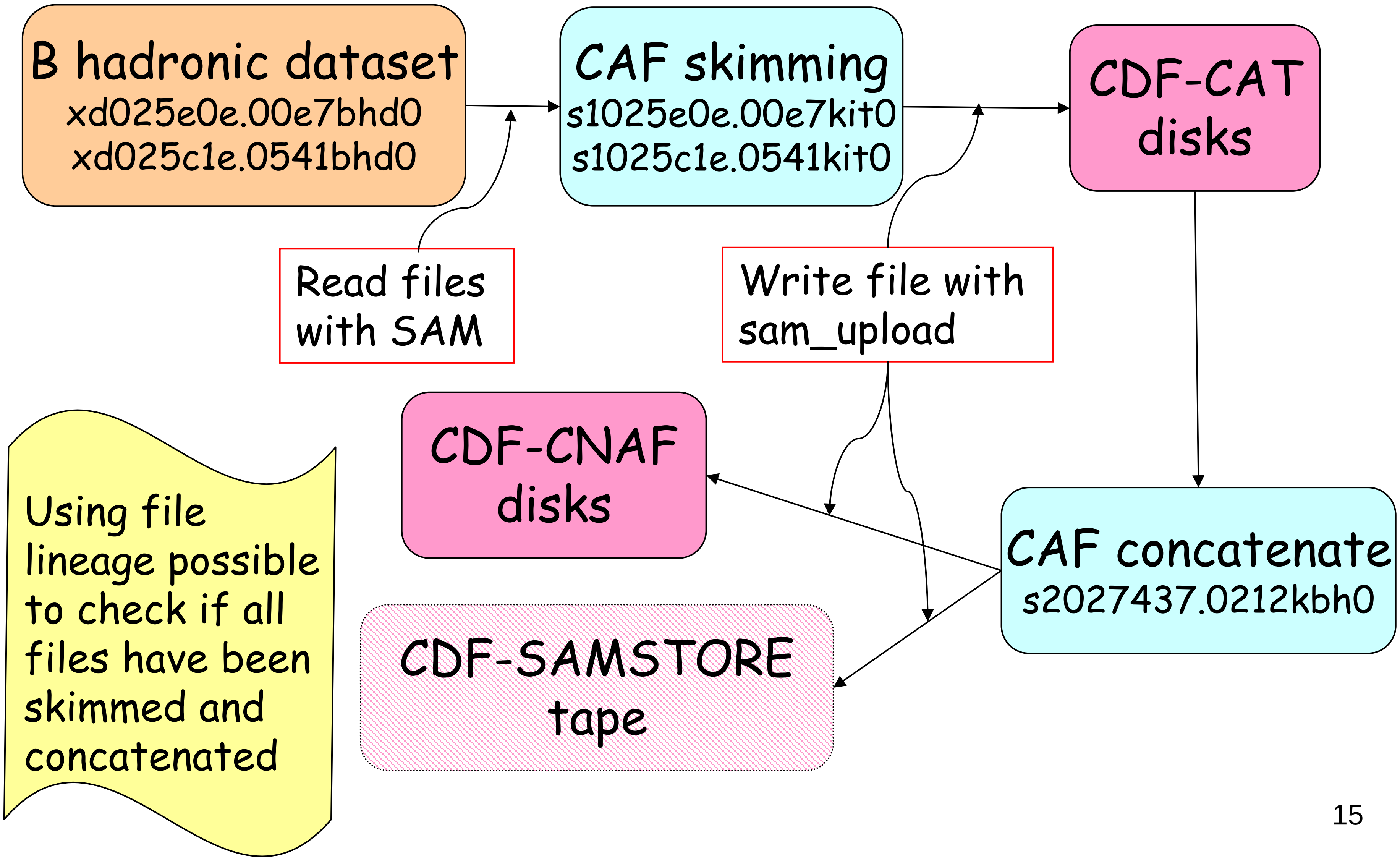
User Registration and SAM station configuration

- To use the sam_upload the user needs to register personal subject to the SAM station GRID mapfile:

```
fcdflnx2:/cdf/home/lucchesi>setup sam_upload v2_0_13
fcdflnx2:/cdf/home/lucchesi>kx509
fcdflnx2:/cdf/home/lucchesi>kxlist -p
Service kx509/certificate
 issuer=/DC=gov/DC=fnal/O=Fermilab/OU=Certificate/
 Authorities/CN=Kerberized CA
 subject= /DC=gov/DC=fnal/O=Fermilab/OU=\
 People/CN=Donatella Lucchesi/0.9.2342.19200300.100.1.1=lucchesi
 serial=03AD9E
  hash=8a818628
```

- SAM station has to be properly configured: 2 stations at fnal (cdf-cat, cdf-samstore) and one in Italy, cdf-cnaf.
- Policies to decide who can write to tape and to common disks have to be decided by the physics, offline and data handling group

Sam_upload use case: hadronic B skimming



Summary

- ✓ The necessary tools to store files on SAM are in place
- ✓ B hadronic dataset skimming is the first stress test done. Scalability issues have been found. With a lot of work of the CDF SAM team and CD now with the new DB server version the problem seems under control.

Two more comments:

- ➢ Data store with SAM not accessible with DFC: not a problem for remote sites where data access is only via SAM.
  Issue at fnal?
- ➢ At CNAF there is the full BCHARM sample data access via SAM for standard CAF and GlideCaf succefully since a while.

Backup

CDF Autodestination Server

- Enstore file family:
  - Set of files that should be compact on tape
    - CDF convention: “file family” = “CDF dataset”
- CDF Enstore PNFS structure is like:
  - /pnfs/cdfen/filesets/SM/SM02/SM0270/SM0270.4/myfile
  - Every low level directory contains files of one file family
  - Every low level directory contains ~10 files
- Autodestination server
  - Forking TCP connection server
  - Keeps track of existing directories for different datasets
  - Creates new directories as necessary, set proper file family
  - Declares newly created directories to SAM