Data Handling: SAM update; file access via XRootD; StashCache

Robert Illingworth

FIFE workshop 2017

21 June 2017

Data handling

- Data handling is your interface with the data
  - Crucial part of all production and analysis chains
- Cornucopia of strangely named technologies…
  - dCache, EOS, NFS, BlueArc, gridftp, http, xrootd, StashCache, SAM, IFDH, FTS, a different FTS, etc
  - Time and technology marches on, and we sometimes have to change the way storage access works (for example, BlueArc unmounting)
  - Using common tools helps to abstract away some of the differences and complications and eases some of the pain of transitions

Data handling

- This isn't a general introduction to the Data Handling tools
  - See for example the 2015 FIFE workshop talk for that https://indico.fnal.gov/contributionDisplay.py?contribId=35&confId=9737

- Also see the best practices talks for Data Management and SAM4Users tomorrow

- The rest of this talk covers things that have changed, some things we want to promote more use of, and some thoughts and plans for the future

SAM update

- Recent changes are evolution, not revolution
  - Performance enhancements
  - Easier way to enable admin access
  - Better integration with POMS
  - New query operators: intersect, isancestorof, isdescendantof

- I will give more details for a few of these changes since some of them are easy to miss and many may not know about them

SAM performance improvements

- Mostly internal and unnoticed
- Some restructuring of parameter DB tables to speed them up
  - Partitioned NOvA parameter tables to speed it up even more
    - This was a lot of work: we don't plan to do this for any one else unless it's clearly necessary
    - Transparent to end users
- Still scope for improving scalability of some aspects
  - Max number of simultaneous projects is currently memory limited
- Unfortunately some parts are constrained by past design decisions and would be difficult and disruptive to change

Admin access

- You can now grant admin access to another user by adding them to the `admin_role` group
  - Obviously you have to be an admin already to do this
  - The config file still takes precedence (so no one can revoke my access!)

- No need to make a ticket to add new admins as long as there's an existing admin available

- (Maybe we will eventually get this sort of info from FERRY and not need to track it within SAM at all.)

New query operators

- `intersect` set operator
  - Why do you want this – doesn't "`and`" do the same thing? Not always…
  - Useful for locations: otherwise you can't easily find files which are available at all of a set of locations. `and` doesn't work for this; it always returns nothing.

- `isancestorof/isdescendantof`
  - Like parent and child, only works through multiple levels
  - Potential performance issues – stick to simple queries
    - We've blocked the use of more than one of these plus or combined with any other lineage operator in the same query

IFDHC changes

- IFDHC v2 released (now up to v2_0_8)
- Major rewrite of `ifdh cp` internals
  - Largely configuration file driven, with less hardcoded logic
  - Makes it easier to modify behavior – don’t necessarily need to recompile

- Started to phase out cpn locking
  - Not needed where BlueArc is not mounted on worker nodes
- Reducing cases that use gridftp access to BlueArc
  - For when it goes away completely – put your data files in dCache

XRootD

- XRootD is yet another file transfer protocol
- Built in support in root, and optimized for streaming data
  - Send the data to the job without putting it on the local disk

- This is especially beneficial if you aren’t reading the entire file (skip events or branches)

- But even reading the whole file can benefit: you aren’t competing with all the other jobs for the disk

How to use xrootd access

- Direct access to file catalogued in SAM:

```
$ samweb get-file-access-url --schema=root NoiseRun-
0001227-00041.root
```

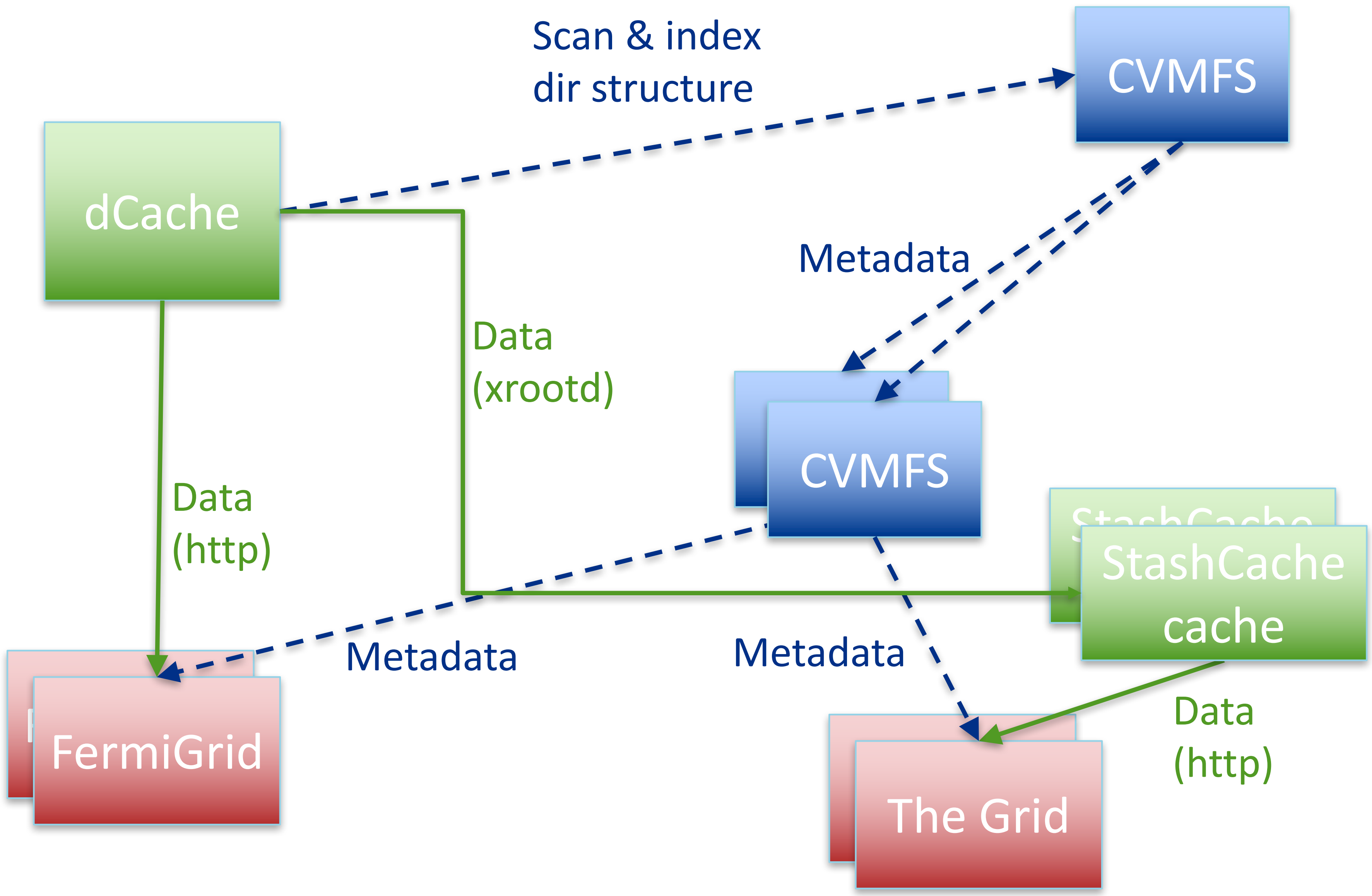
```
root://fndca1.fnal.gov:1094/pnfs/fnal.gov/usr/uboone/
data/uboone/raw/Swizzler/swizzle/v04_18_00/00/00/12/2
7/NoiseRun-0001227-00041.root
```

- In SAM project pass root to the `schemas` option to establish-process (or the last argument of ifdh establishProcess).
  - ifdh art will pass root:// urls through to open them directly, not copy the data locally

StashCache

- StashCache is a system for distributing auxiliary files to batch jobs.
  - Intended for up to TB scale datasets, especially those that get a large amount of use and will benefit from caching, for example flux files
  - Combination of dCache, xrootd and cvmfs.
  - Interface is cvmfs, so just open the file as if it was a local path
  - Utilizes OSG regional caches, so jobs on the grid can access data from a closer source than FNAL

StashCache architecture



Fermilab

StashCache deployment

- Currently available for NOvA, uboone, mu2e and DES

- Unfortunately, not all sites run a compatible cvmfs yet:
  - You can restrict jobs to suitable sites and a minimum repo revsion with something like
  - `--append_condor_requirements='(TARGET.HAS_CVMFS_NOVA_OSGATORAGE_ORG==\"true\"&&TARGET.CVMFS_NOVA_OSGSTORAGE_ORG_REVISION>1234)'`

- If other experiments would like to use this, please make a request through the servicedesk

Future projects

- The following slides describe some future changes – some implemented, some in progress, and some under consideration
  – This is of course in addition to bug fixes, performance and scalability changes, and minor features

Improve handling of restarts in projects

- The current handling of restarted (preempted) batch jobs is not ideal
  - Files consumed by a SAM process are marked as used
  - The job is killed and rerun, losing the output
  - SAM treats the restarted job as totally separate and gives it new files from the project

- The change would be to detect restarted jobs – they have the same ID as a previously seen job.
  - If we see one of those, mark the previous job as bad, and "unwind" the consumed file states so they can be given out again. The restarted job won't necessarily get the same files, but some job should see them

Reduce snapshot churn

- Currently every time you request a snapshot (including starting a project with a definition name) the definition is reevaluated and if it’s changed a new snapshot is added to the database
  - This can be slow if the definition is complex
  - Some workloads can create a lot of snapshots if new matching files are continually being added
- Considering adding feature that if a snapshot has been evaluated “recently”, we immediately return the last snapshot instead of evaluating the definition
  - How recently? 1 hour?
- Some workloads may require forcing the very latest every time, so there would be a way to override this

Upcoming FTS improvements

- Improved checksum support
  - Add support for calculating checksums using a variety of algorithms (enstore, adler32, plus anything supported by python hashlib and crcmod packages)
- Use xrootd as a “local” disk
  - Adding support for using xrootd as the input dropbox location
  - Allows other storage systems like EOS
- Integrate with the LCG FTS3 file transfer manager
  - I know: it’s confusing terminology
  - Should provide more efficient, self tuning, WAN file transfers

Centralized output path generation

- Currently upload paths are set in the F-FTS config file (or in sam4users scripts, or by hand)
  - Makes it harder to be consistent across systems

- Add storing templates for the trailing part of the path in the central DB
  - Simple metadata rule based system with syntax similar to queries, for example to match raw files:
    - `data_tier raw :   raw/${run_number}/${data_stream}`
  - Combine with site prefix to get the full path
  - Potentially improve FTS performance and memory footprint since it would no longer need to download all the metadata

Dataset transfer service

- This would be a service that would allow you to request making a dataset available a given site and it would automatically go off and do this
- Still in early prototyping/design stage
- Can’t become a real project until we come up with a proper name!

The longer term future

- What should the data handling system of the future look like?
  - What parts of the current system do we want to retain?
  - What parts should be jettisoned?
  - What has to be abandoned because it can’t scale?
  - Can we benefit from tools and systems developed for HL-LHC?

- At the moment we have more questions than answers
  - The future is nearer than it looks!

Final thoughts

- Development is driven by a combination of feedback from users, anticipation of future needs, and available resources.

- We're always interested in hearing comments and suggestions for improvements
  - Sometimes what's requested may already be possible
  - Sometimes it can be added with relatively little effort
  - Sometimes more significant features may be added, but it's a high bar
  - (Sometimes it's a good idea, but just not practical to implement)

- A major driver at the moment is to improve distributed data access, with HEPCloud on the horizon