StorageD

A “customisable workflow”

Kevin O’Neill and Roger Downing
STFC eScience

Science & Technology Facilities Council

StorageD

- Originated to take experimental data from Diamond central storage to the tape store interface
  - Component of the STFC eScience framework for facilities
  - In production in Diamond since 2009
  - Integrates with other components used by Diamond such as ICAT
  - Archived over 155Tb (>36.7m files) of experimental data in v1
  - V1 developed and proved the overall workflow
  - V2 stored 262Tb in ~113m files since July 2011
  - Deployed in CEDA (currently >366Tb in >9.5m files)

Science & Technology Facilities Council

StorageD provides…

- A pipeline from the user's data area to the end data resource (CASTOR) with:
  - Validation of data files – checks that files:
    - have correct permissions and filename syntax;
    - are available
    - have not already been archived
  - Packaging of data files into "retrieval-friendly" aggregations (with checksums)
    - Aggregation mechanism takes into account expected method of retrieval.
  - Back-end store transparency to the end user:
    - SRB/ADS (old)
    - CASTOR (current)
    - Mounted file systems
  - Monitoring of the data movement process
    - Error reporting, customised to requirements
    - Real time web-based monitoring

Design criteria

- Transport and store large-volumes of data according to a defined, rarely changing, workflow
- Efficient
- Maintainable
- Easily adaptable and extensible
- Progress of files through the system is fully:
  - Verified
  - Tracked
  - Auditable

StorageD performance and scaling

- We've found…
  - Two axes affecting performance:
    - Registration and package definition:
      - can handle well over 1m files per hour, dependent on factors such as the file system
    - Creation and movement of packages to end storage (aka volume)
      - ability to allow as many registration and transfer processes as required means that bottle neck lies in client file system, network and ability of storage resource interface to accept input

Aggregations

(aka transfers, containers…)

- Bring together files into units for:
  - Transfer across the network
  - Storage in tape system
- Default method is concatenation
  - But “tarring” tested and others possible
- Currently, only one aggregation method per installation
- Assembled according to “rules”

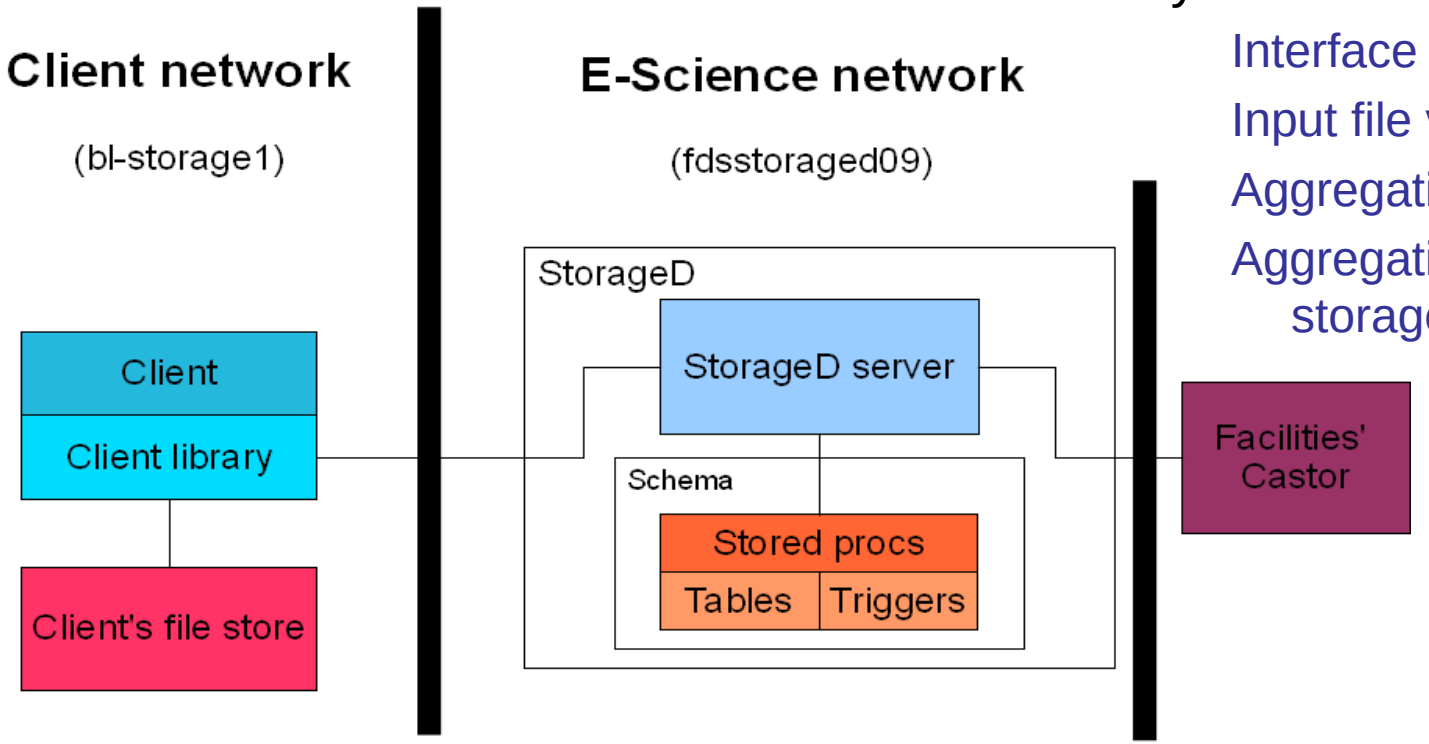
Aggregation rules

- Files allocated by rule
  - Files from a beamline/visit (DLS) or dataset/management unit (CEDA) allocated to an open aggregation, if none then create one and populate
- Closure by rule
  - Currently set per beamline
  - Simple closure parameters held in DB table
    - Max size (usually ~5GB for DLS, ~10GB for CEDA)
    - Max number of files
    - Max time no activity (30 mins norm for DLS) trumps all

General architecture

Python code provides:

- Interface to file store(s)
- Input file validation
- Aggregation instantiation
- Aggregation transfer to storage resource



Database provides:

- Process control
- File/aggregation information
- Allocation of files to aggregation
- Auditing of process

Science & Technology Facilities Council

DB Interface

- Input via bulk ingest utility
  - SQL*Loader for Oracle
- Series of stored procedures called by code to perform defined task

add_batch_log
add_error
get_aggrgtin_dtls_by_trnsfr_id
get_file_list_by_transfer_id
get_file_details_by_file_name
get_transfer_id_next_flushable
get_transfer_id_next_syncable
get_transfer_id_next_trnsfrble
set_transfer_end_copy
set_transfer_end_sync
set_transfer_end_flush
set_transfer_error_copy
set_transfer_error_flush
set_transfer_error_retrieve
set_transfer_error_sync
set_transfer_start_retrieval
set_transfer_end_retrieval
set_trnsfr_md5sum_by_trnsfr_id

Science & Technology Facilities Council

“User Interface(s)”

- Basically a file list:
  - Tell us where to get the files to store, and we’ll get them (“terms & conditions apply”)
  - For Diamond, location extracted from ICAT XML file
  - For CEDA, an enhanced file list will be used, but currently just a simple list
- However, custom registration clients written against the APIs are possible

Science & Technology Facilities Council

State-driven transfers…

- Each transition logged
  - CREATION — Initial state when first file about to be assigned to it
  - BUILDING — Accepting files
  - CLOSING — Transfer is about to close, but needs closure procedures running on it
  - CLOSED — Transfer is closed to new input
  - TRANSFERABLE — Aggregation closed and ready for actual transfer
  - COPYING — Copy to StorageD disk cache in progress
  - CACHED — Copy to StorageD disk cache completed
  - CACHED_SYNCED — Aggregation stored on disk and storage resource (CASTOR)
  - FLUSHING — Copy of aggregation being removed from StorageD cache
  - SYNCED — Aggregation stored on CASTOR
  - DELETED — Aggregation deleted for some reason. See history and error table
- And accompanying error states
- Where necessary, states require that an aggregation is “owned” by a process/server

All logged in “loving detail”

Science & Technology Facilities Council

Getting it back…

- Two main ways
  - Custom interface talking to retrieval APIs
  - A line mode command (sd_get) which can take a file list or a “high level aggregation”, such as a visit ID.
- Retrieval request created that then:
  - DB “finds” the necessary aggregations to be retrieved, “locking” any already on cache
  - Code recalls aggregations to cache (if necessary)
  - desired files extracted and delivers to the client as they are available
  - State of all elements in tracked throughout the process

Futures…

- So many possible…
  - Improved consistency checking,, such as file level checksumming
  - Improved cache communication
  - More flexible “aggregation ownership” in the workflow
  - More complex “aggregation rule” processing
  - Cover a wider range of scenarios for multiple servers and clients
  - Wide Area StorageD
  - Etc, etc…

Thank you…

For more details, contact:

Roger Downing (roger.downing@stfc.ac.uk)

Kevin O'Neill (kevin.o'neill@stfc.ac.uk)

Science & Technology Facilities Council