WAN dCache transfers

FIFE meeting, 02/17/2022

Dmitry Litvintsev

Switch to HTTP(s)

- Observe the following errors on dCache end:
  1. `Channel idle for too long during upload`. Ends up with broken file replica in dCache.
  2. `No connection from client after 300 seconds. Giving up`. Nothing uploaded.
  3. `Connection lost before end of file`. Ends up with broken file replica in dCache.
- Almost all errors from
  - `192.12.238.129 (gridftp000.colorado.edu)`
  - `137.99.174.0 subnet (uconnecticut)`

Issues

- All errors listed result in failed transfers that client (`ifdh cp`) tries to handle by retrying.
  - But if destination exists (broken files) it fails on retry as destination already exists and we have a policy of not overwriting destination if it already exists (this is matter of configuration).
- Result:
  - dCache accumulates broken files
- Attempts to open broken files via NFS lead to processes to be in D-state (non-interruptible I/O). The only cure – removing the files.
  - FTS that handles uploaded files hangs without much progress on processing ingest (issue first reported by GM2)

Remedies

- Suggested to switch back from HTTP to GFTP until we figure out solutions.
  - Switched back to GFTP on 02/10 (I think).
- dCache upstream:
  - NFS server already fixed not to hang on broken files (patch needs to be deployed).
  - A couple of patches made to WebDAV to try to detect and remove targets results from failed transfers (needs to be carefully tested and deployed).

Transfers over WAN

- Looking at broken files issue brought to the fore considerations of end-to-end data integrity when transferring TiBs of data over the WAN.
- Yes, in this case broken files are “artificial”, should not have been left behind and transfers did error out. So, in principle, these files should be known to client to be bad. It is just not easy to relay this info to other processes that monitor and handle incoming data (like FTS).
- What if file is corrupted while being transferred and transfer is successful. Remember dCache just gets a stream of bytes and calculates checksum to be stored in namespace. It needs to compare it with something to guarantee end-to-end integrity!

How integrity generally handled

- LHC moves data using srm copy and it has provision to calculate and compare local checksum with checksum calculated on the server. This is ”on-the-flight” checksumming.
- In addition, if checksum is already known, it can be passed along for comparison with checksum calculated by the server.
- Or file can be transferred, its checksum queried after transfer and then compared with original local checksum by the client “manually”
- Having looked at www_cp.sh I realized that IFDH does none of the above.
- So how can it be corrected (if agreed that this needs to be done).

Gfal?

- www_cp.sh uses `curl` to transfer files. Currently there is no provision to pass adler32 checksum using curl. One can use MD5 for instance by passing `Content-MD5` header. dCache then will calculate MD5 in addition to adler32. Multiple checksums may confuse backend HSM script so needs to be tested.
- Other clients – `xrdcp, globus-url-copy, srmcp` support adler32. But these options do not seem to be used by `ifdh cp` (I may be wrong)
- `ifdh cp` seems to me to be a wrapper tool that calls different clients based on protocol specified.
- CERN has similar product – GFAL (grif file access library) that provides command line tools like `gfal-copy` and `gfal-sum`
- `gfal-copy` is a uniform client that can use any supported protocol. So seems similar to `ifdh cp` in functionality.
- For HTTPs transfers `gfal-copy` uses `davix` library which in turn uses `neon` WebDAV client library (so does not use `curl`). It supports adler32 checksumming.