Table of Contents

CASTOR Notes

Service classes

A service class corresponds to a SRM space token.

The relation among service classes and disk pools may be many-to-many. In practice:

- at CERN, it is 1-1 for the LHC experiments
- at CERN and some external sites, it is N-to-1 for smaller VOs which share the same disk pool.

Each service class has its own Garbage Collector (GC), hence in a shared disk pool scenario, there is room for interference among different VOs. This also constraints the service classes sharing the same disk pool to have the same retention policy.

Space management

The smallest entity with an associated amount of total and free space is the disk pool.

xrootd

The xrtood interface is greatly improved from CASTOR 2.1.8, and is currently accessible via the CMST3 service class. This class is shared between ATLAS and CMS and is intended for user analysis.

CMS foresees to use xrootd for analysis activities at CERN on this dedicated service class. If things go well, it may eventually be extended to all analysis activities (i.e. the CAF) and even T0 processing. Using xrtood will be possible only with new versions of CMSSW.

Bugs

Discovered a memory leak in the SRM server (bug #107344).

The internal GridFTP is incompatible with the dCache Bestman clients because they require at least two GridFTP sessions, while the CASTOR TURL becomes invalid after one session. It is agreed that CASTOR is violating the SRM specification about the TURL having to be valid until the pin expires. No changes in CASTOR are planned for the moment.

Pin lifetime

Pinning in Castor is 'advisory', meaning that depending on the back-end service policies, it may well be completely ignored. In other words, the only way to get 'hard pinning' is to use a Disk1 storage class, where copies remain on disk until users explicitly delete them via srmPurgeFromSpace, whereas a Tape1Disk0 storage class won't offer any guaranteed lifetime for TURLs as the GC is in charge, thus making the srmReleaseFile request basically unneeded (from a mail of Giuseppe).

Specifying a pin lifetime in fact contributes a term in the garbage collector weight for the file, and any lifetime longer than 30 minutes is capped at 30 minutes. So, the influence is rather small.

srmReleaseFiles has no effect in CASTOR, while srmPurgeFromSpace effectively removes a file from a disk pool, provided that another copy of the file exists.

Checksum

With CASTOR 2.1.8, if a checksum is preset before a transfer to CASTOR, CASTOR will check that the actual checksum is consistent.

Request lifetime

What follows applies for sure to a BringOnline request, but probably also to PrepareToPut and PrepareToGet requests.

When SRM polls the backend to update the status of a request, in case of no changes it backs off after an exponentially increasing time. When the back off time exceeds 10 hours, a last poll is done and the request ends *(with which status?)*. After 10 days, the request token disappears from the SRM database.

Any specified *desiredTotalRequestTime* is in fact ignored, as there would be no gain in aborting a request after that time, considering that a stage request to CASTOR cannot be aborted. *(Check if a totalRequestTime is returned)*

Installed versions of CASTOR (04/08/2009)

- CNAF: 2.1.7-17 now, 2.1.7-27 by the end of the week
- RAL: 2.1.6-27?
- ASGC: 2.1.7-24, latest SRM version

The "big ID" bug has been fixed in CASTOR 2.1.7-27 and in the latest SRM version. The workaround is not present in SRM 2.8 because there is a proper patch for Oracle.

-- AndreaSciaba - 02 Mar 2009

This topic: LCG > WlcgStorageCastorNotes
Topic revision: r4 - 2009-08-04 - AndreaSciaba