Distributed OCDB using CVMFS

Raffaele Grosso

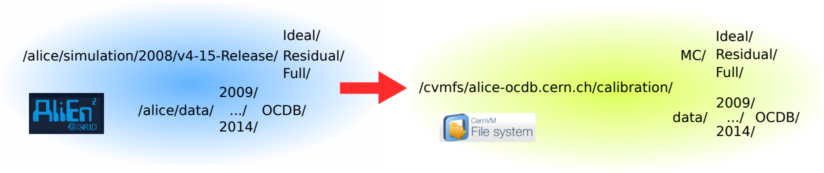
November 19, 2014

Intro 1: OCDBs

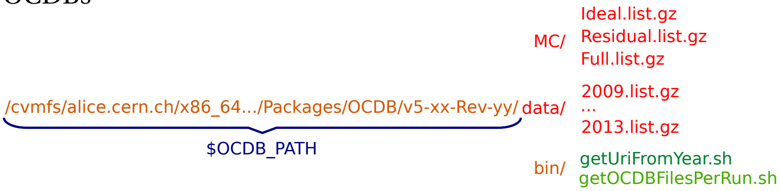
- Producers of OCDB objects are:
  1. the Shuttle (automatic upload)
  2. CPasses (automatic upload)
  3. manual uploads, requested via JIRA tickets
- Consumers of OCDB objects are:
  1. the Shuttle
  2. raw, MC productions
  3. analysis trains
  4. single users

Intro 2: OCDBs on CVMFS

1. OCDBs on CVMFS are a synchronized copy of the AliEn OCDBs;



2. CVMFS OCDB packages provide a way to see a “frozen” picture of the OCDBs



Intro 2: OCDBs on CVMFS

- Two purposes:
  1. avoid clashing with OCDB uploads
  2. store the information of the status of the OCDBs in a single place for an entire production

Same use as the per-run OCDB snapshots already in use. But more practical and more general: being instead full OCDB snapshots, they allow "OCDB versioning" and "OCDB tagging" (see later).

Synchronization

- In the initial implementation, synchronization between AliEn and CVMFS OCDBs has been done “committing” to CVMFS after manual uploads (code in AliRoot), while Shuttle and CPasses’ OCDB objects were synchronized on a non-regular basis.
- Synchronization is now centralized, provided by an AliEn optimizer (Miguel’s work)
- All OCDB objects are “committed” to CVMFS practically at the moment they are uploaded to AliEn
- The time for them to be visibile in CVMFS is the time to be propagated to CVMFS strata.

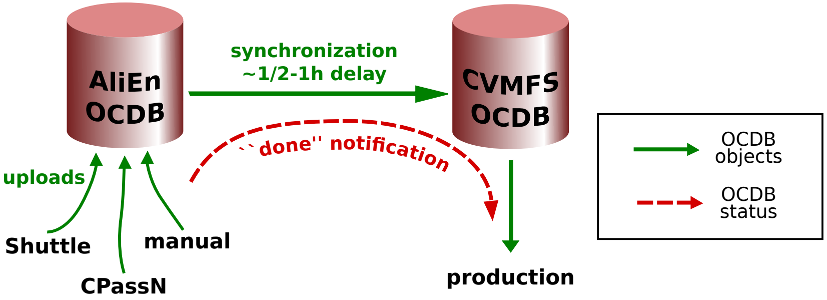
Full OCDB snapshots

- The user (production) sees the OCDBs as they were at a given time, unaware of later uploads, by:
  1. loading a CVMFS package including the list of OCDB files at that given time ("full OCDB snapshot") and
  2. setting an environment variable ($OCDB_PATH) which instructs the CDB framework to look at the snapshot when querying for an OCDB object
- Flexibility in the workflow:
  - When the user (production) uploads an AliRoot package, it can choose which OCDB package to upload with it (AliRoot packages are not bound to specific OCDB snapshots)
  - It is technically possible, although not advisable, to retrofit CDB objects by manually editing the OCDB objects list before packaging it

What is missing?

We need to tag and propagate the status of the OCDB from the producers (see slide 1) to the consumers for:

1. timely publishing the full shapshots (CvmFS OCDB packages)
2. allow productions to check their requirements agains the OCDB snapshots



BTW, finding a reliable solution to this would also improve our current (manual) workflow.

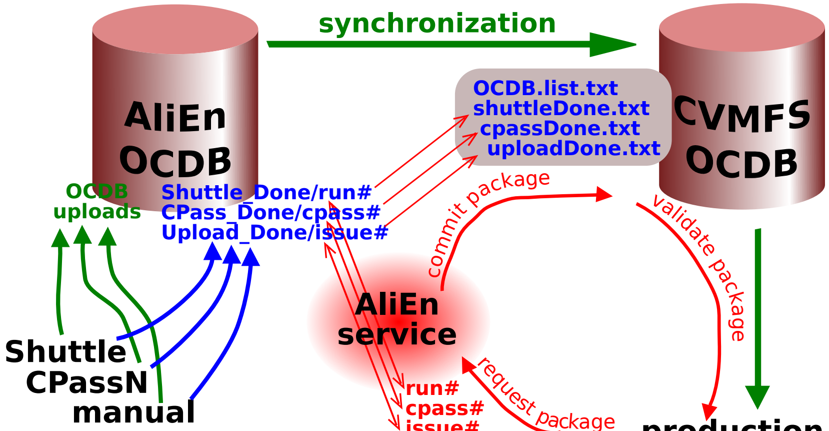
Implementation

CvmFS OCDB packages versioning and requests:

- All producers of OCDB objects touch a file in AliEn to mark that OCDB objects are done:
  1. the Shuttle already touches `/alice/data/20xx/SHUTTLE_DONE/runNumber`
  2. CPasses include in their validation the touching of `/alice/data/year/CPasses_Done/CPassId`
  3. manual uploads are followed by touching say `/alice/data/year/JiraDone/TicketNumber`
- A dedicated AliEn service (maybe a new version of the current optimizer) compares production requests against those "done" files. When the requests are satisfied, the service it commits a CvmFS OCDB package, containing
  1 files with a list of "done" files:
    - 2 Shuttle done
    - 3 CPass done
    - 4 Jira uploads done

  5 a file with the list of OCDB files
- Productions read this tag to check that the CvmFS OCDB package

Implementation


synchronization
AliEn OCDB
CVMFS OCDB
OCDB.list.txt
shuttleDone.txt
cpassDone.txt
uploadDone.txt
OCDB uploads
Shuttle_Done/run#
CPass_Done/cpass#
Upload_Done/issue#
Shuttle
CPassN
manual
AliEn service
run#
cpass#
issue#
commit package
validate package
request package
production