OBSMON

Observation monitoring of a NWP run

Trygve Aspelien & Roger Randriamampianina

-- Data assimilation aims at building a best possible atmospheric state using observations and short range forecasts.

Best Linear Unbiased Estimator (BLUE) analysis:

$$x_a = x_b + K[y_o - H(x_b)] \qquad K = BH^T(HBH^T + R)^{-1}$$

We usually choose the following parameters/outputs for monitoring:

- fg_depar (in ODB): y_0 – $H(x_b)$: innovation or first-guess departure – in observation space
- an_depart (in ODB): y_0 – $H(x_a)$: analysis departure
- Observation usage status (spatial and temporal)

Concepts:

- 3/4D-VAR and CANARI use observations and stored in ODB format
- We want a flexible graphical tool to see how observations were used
  - Should be possible to use offline or as a server
  - Visualize on demand instead of pre-generate all plots

Two building blocks

1. Extraction of ODB to SQLite database
    a. Fortran program with SQLite binding
    b. Possible to monitor individual observations
    c. Externalized in cy43
2. R + package shiny
    a. https://git.hirlam.org/Obsmon

In obsmon a query for conventional data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $press1=-1;
SET $press2=-1;
SET subtype1=-1;
SET subtype2=-1;

CREATE VIEW obsmon_conv
SELECT
obstype@hdr,codetype@hdr,statid,varno,lat@hdr,lon@hdr,vertco_type@body,vertco_reference_1@body,sensor@hdr,date
,time@hdr,report_status.active@hdr,report_status.blacklisted@hdr,report_status.passive@hdr,report_status.rejected@hdr,
datum_status.active@body,datum_status.blacklisted@body,datum_status.passive@body,datum_status.rejected@body,dat
um_anflag.final,an_depar,fg_depar,obsvalue,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( codetype@hdr >= $subtype1 ) AND
( codetype@hdr <= $subtype2 ) AND
( vertco_reference_1@body > $press1 ) AND
( vertco_reference_1@body <= $press2 ) AND
( an_depar IS NOT NULL )
```

In obsmon a query for conventional data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $press1=-1;
SET $press2=-1;
SET subtype1=-1;
SET subtype2=-1;

CREATE VIEW obsmon_conv2
SELECT
obstype@hdr,codetype@hdr,statid,varno,lat@hdr,lon@hdr,vertco_type@body,vertco_reference_2@body,sensor@hdr,date
,time@hdr,report_status.active@hdr,report_status.blacklisted@hdr,report_status.passive@hdr,report_status.rejected@hdr,
datum_status.active@body,datum_status.blacklisted@body,datum_status.passive@body,datum_status.rejected@body,dat
um_anflag.final,an_depar,fg_depar,obsvalue,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( codetype@hdr >= $subtype1 ) AND
( codetype@hdr <= $subtype2 ) AND
( vertco_reference_2@body > $press1 ) AND
( vertco_reference_2@body <= $press2 ) AND
( an_depar IS NOT NULL )
```

In obsmon a query for satellite data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $sensor=-1;
SET $press=-1;
SET $statid=-1;

CREATE VIEW obsmon_sat
SELECT
obstype@hdr,codetype@hdr,satellite_identifier,varno,lat@hdr,lon@hdr,vertco_type@body,
vertco_reference_1@body,sensor@hdr,date,time@hdr,report_status.active@hdr,report_status.
blacklisted@hdr,report_status.passive@hdr,report_status.rejected@hdr,datum_status.active@body,
datum_status.blacklisted@body,datum_status.passive@body,datum_status.rejected@body,
datum_anflag.final,an_depar,fg_depar,obsvalue,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat,sat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( sensor@hdr = $sensor ) AND
( vertco_reference_1@body = $press ) AND
( satellite_identifier = $statid ) AND
( an_depar IS NOT NULL )
```

queryname=”obsmon_sat”, “obsmon_conv” or “obsmon_conv2”

```
iobs=0
WRITE(*,'(A,I4,A,I4,A,A)') ' Monitoring obs #',obs,'/',nused,' Exec query="'//trim(queryname)//'"'
  rc = ODB_select(h,queryname,nrows,ncols,nra=nra,poolno=-1,&
                  setvars=varnames,values=vars)
  IF (nrows > 0) THEN
  ALLOCATE(x(nra,0:ncols))
  rc = ODB_get(h,queryname,x,nrows,ncols,poolno=-1)
  DO jr=1,nrows

    !write(*,'(1p,(5x,10(1x,g10.2)))') x(jr,1:ncols)

    IF ( ncols /= 26 ) THEN
      WRITE(*,*) ' Inconsistency in NCOLS found and expected:',ncols
      CALL abort
    ENDIF
    ! Assign odb extract to module variables
    iobtyp_odb(1)=int(x(jr,1))
    icodetype_odb(1)=int(x(jr,2))
    !WRITE(cstaid_odb(1),'(F8.0)') x(jr,3)
    WRITE(cstaid_odb(1),FMT=pstring%fmt) x(jr,3)
    ivarno_odb(1)=int(x(jr,4))
    rlat_odb(1)=x(jr,5)
    rlon_odb(1)=x(jr,6)
    IF ( lradians ) THEN
      rlat_odb(1)=180.*rlat_odb(1)/pi
      rlon_odb(1)=180.*rlon_odb(1)/pi
    ENDIF
    ivertco_type_odb(1)=int(x(jr,7))
    rpress_odb(1)=x(jr,8)
    isensor_odb(1)=int(x(jr,9))
    idate_odb(1)=int(x(jr,10))
    itime_odb(1)=int(x(jr,11))
    istatus_acthdr_odb(1)=int(x(jr,12))
    istatus_blkhdr_odb(1)=int(x(jr,13))
    istatus_pashdr_odb(1)=int(x(jr,14))
    istatus_rejhdr_odb(1)=int(x(jr,15))
    istatus_actbod_odb(1)=int(x(jr,16))
    istatus_blkbod_odb(1)=int(x(jr,17))
    istatus_pasbod_odb(1)=int(x(jr,18))
```

(offline) scripts for extraction from ODB

https://git.hirlam.org/Harmonie/util/obsmon/scr

Create SQLite tables pr task:

- obsmon_stat_all DTG odbpath ecma/ccma -c config [-p np] [-g] [-a ARCHIVE] [-m]
  - config ~ scr/include.ass
  - np = tasks in parallell
  - m activates multitask
  - g (ground) monitors CANARI (ecma_sfc)

Create one SQLite database for this ODB:

- obsmon_link_stat_all $DTG ecma/ccma -c $config [-g] -a ARCHIVE

scripts for extraction from ODB in HARMONIE

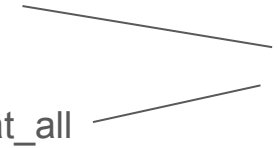
MULTITASK=yes/no

yes: One task running on more nodes distributing the task.

- Listening to a signal file

no: One task in scheduler for each observation type

- obsmon_stat
  - -> obsmon_stat_all
- obsmon_link_stat
  - -> obsmon_link_stat_all



Same scripts as the offline method on the previous slide!

Files

- Input ODBs:
  - $HM_DATA/YYYYMMDD_HH
    - odbvar
    - odb_ccma
    - odb_can

- Output
  - $EXTRARCH -> $ARCHIVE_ROOT/extract -> $HM_DATA/archive/extract
    - ecma/YYYYMMDDHH/ecma.db
    - ccma/YYYYMMDDHH/ccma.db
    - ecma_sfc/YYYYMMDDHH/ecma.db

(1) Future: ODB-API

- Create a flat file in ODB2 from ODB1 for archiving (odb_migrator)
- Use the python interface to ODB-API to select from ODB and create an SQLite database
  - Pros:
    - Can more easily be (re)-done at any time
    - Don’t need a hacked SQLite binding to Fortran around SQLite
    - Can replace all shell scripts with native object-oriented python code with built-in SQLite support
  - Cons:
    - Still need an extra database as long as ODB-API does not support R or as long as the visualization tool is written in R (using shiny). This could also be done in a similar python framework (e.g. bokeh).

(2) Shiny

1. Get obsmon from hirlam.org:
   git clone https://git.hirlam.org/Obsmon obsmon
2. Install obsmon:
   cd obsmon
   ./install --local-install
3. Set up a valid config.toml file.
   This file tells obsmon where to find the experiments. Please take a look at the example file "config.toml.example" included with obsmon.
4. Finally, run obsmon:
   ./obsmon --launch