Using the Conditions Database in LHCb
Ph.Charpentier, LHCb SW week, May 2005

Outline:

- Introduction: some definitions
- Online usage
- Offline usage
- Hardware infrastructure

Definitions

- Conditions Database
  - Contains all information that has a time dependence (interval of validity) and is necessary to perform reconstruction or analysis of data
  - Examples:
    - Alignment and calibration constants
    - Some online configuration parameters
    - Ambient parameters relevant for reconstruction (pressure in RICH, presence of HV on detectors)
    - Some machine parameters (e.g. energy, luminosity…)
    - Not present: current in chambers, individual bunch currents, temperature and pressure probes (unless relevant for reconstruction)…

Definitions (cont'd)

- Configuration Database (see Clara's talk)
  - Used by the Online system in order to configure the whole system
  - Readout configuration, electronics boards parameters, FPGA programs, pedestals etc…
  - For the EFF (relevant for the ConditionsDB)
    - version of applications to be used
    - Run-time configuration of parameters ("jobOptions")
    - Tag of the ConditionsDB to be used
- Online PVSS archive
  - Contains all parameters controlled by the ECS
  - The memory of the detector

Sources and Types of Conditions

- Sources
  - The Online system (see Clara's talk)
    - Configuration parameters, hardware readings, DCS parameters…
    - Few Online alignment/calibration constants (e.g. VELO x positions)
  - Offline alignment and calibration
    - Obtained from offline algorithms: internal alignment, global alignment, calorimeter calibration, corrections etc…
- Types
  - Online Conditions
    - COOL Single-version folders
  - Offline Conditions
    - All others, even if produced in the Online environment

Online Usage

- Conditions are used by applications running in the EFF
  - L1, HLT & Brunel
  - Reproducibility: L1 and HLT algorithms must be possible to be re-run offline using the same Conditions as Online
    - No such constraints for Brunel, as only used for hotline analysis
    - L1 and HLT should only use Online Conditions or tagged Offline Conditions
- Uploading Conditions during Configuration of EFF-nodes
  - Offline Conditions: corresponding to a tag (obtained from the ConfigDB)
    - The tag is saved in the ConditionsDB, valid from configuration time
  - Online Conditions: currently valid Conditions are used
    - Note that Conditions might not be recorded while not data taking

Online Conditions update

- Some Online Conditions may change during data taking…
  - If used by Online Applications, they must be uploaded to the EFF
    - If not used, they are not needed!
    - Validation procedure is up to sub-systems (filters in Clara's talk)
  - Proposed procedure
    - If the run is stopped/paused
      - New value stored in ConditionsDB with current time as start of validity
      - Uploaded to the Online CondDB service (on EFF), force update
    - If the run is going
      - Problem with offline reproducibility of Conditions (time not well defined)
      - The assigned validity time is set to CurrentTime+Δt, Condition is stored in the ConditionsDB
      - Uploaded to the Online CondDBAccessSvc with that validity time. Use Gaucho commands to communicate.
      - Cached temporarily in the Online CondDBAccessSvc, used when first event comes after the validity time, triggers update
      - Δt must be larger than the time needed to upload to the whole farm (few seconds?

Online Conditions update: remarks

- It is explicitly assumed that no Offline Conditions need uploading during data taking
  - This rule is true by definition
    - "L1 and HLT should only use Online Conditions or tagged Offline Conditions"
  - If new Offline Conditions are produced and one wants them to be immediately applicable Online: tag them, stop the run, update the ConfigDB with the new tag value, restart a run
    - By experience this is an irrelevant situation…
- Conditions that need different update filters for trigger and offline
  - Example: pressure variation 0.5 mbar for HLT, 0.1 for offline
  - Should be stored in different CondDB folders

Online "jobOptions"

- Remember there are two++ types of jobOptions
  - Application configuration
    - Sequences, algorithms, System required options (node name, SFC address, Controls PC…)
    - Stable during the whole lifetime of the application
      - No need to upload
    - Correspond to a given Configuration version (saved in the ConditionsDB)
  - Algorithm parameters
    - As in Offline applications default values should be in the code
    - Offline: they can be overwritten by jobOptions
    - Online:
      - Defaults can be overwritten by the Configuration (jobOptions)
      - New values can be uploaded during the run but need to be saved as Online Conditions. Use same procedure as for other Online Conditions update.

jobOptions: remarks

- An application Configuration cannot be modified for a running application: Online and Offline (at least for a foreseeable future…)
  - Questions:
    - How to configure an Offline application for a given set of events, not known a priori? Needs first event's time…
    - How to deal Offline with different Online application configurations in a given set of events?
    - No access to the ConfigDB from an Offline application… Hence the configuration itself should be saved, not the version
      - Problem of algorithm version (code)… No way currently to dynamically select at run time
- Algorithm parameters update
  - The Online jobOptions service needs to subscribe to the corresponding Condition (use naming convention)
  - Declare a dependency on that Condition to the Update Manager
    - Overwrites parameter value when valid

Trigger parameter settings

- Might change during running to accommodate changes in luminosity
  - Only every few hours (LHCb notes 2000-008 and -095)
- Possible scenarios
  - Predefined sets of parameters (in ConfigurationsDB)
  - Values computed by Online trigger monitoring algorithms adjusting the bandwidths (new thresholds, prescalings)
- Actions to be taken
  - For L0 thresholds, need to pause the run before changing
  - L1 & HLT: treat as any Online Condition.
    - Set validity date to a $+\Delta t$, store in ConditionsDB and upload to the EFF
- Offline
  - Get parameters from the ConditionsDB to overwrite defaults

Using Offline Conditions in trigger algorithms

- Requirement:
  - Use different (alignment) conditions in a single application (between trigger and reconstruction algorithms)
- Proposal
  - Use different DetectorElements (mandatory)
    - However they should have an identical structure
    - Only difference: dependence on different Conditions
  - Option 1 (preferred) : Use two DetectorDataSvc
    - Two methods in GaudiAlgorithm for service access
    - Use same path in TDS
    - Online, instantiate only one service (no problem if both methods return the same service)
  - Option 2: Use naming convention (e.g. /dd/online/xxx parallel structure to /dd/xxx)

Offline usage of Conditions

- From ConditionsDB, valid at the event time (GPS time)
  - Recording time cannot be used as the same time must be used Online and Offline
- Offline Conditions
  - During production, should be pretty stable throughout the whole file, but not mandatory…
  - For sparse event sets, may need full update at each event…
- Online Conditions
  - May require some updates during processing
    - Warning: events are not in chronological order on files!
      - Alternative: sort them….
    - For events taken around changes of Conditions, might go forth and back
- Reproducibility
  - Ensured by construction as trigger applications only use Online Conditions and tagged Offline Conditions (the tag is an Online Condition itself!)

Online alignment Conditions

- Some alignment constants may be obtained Online
  - Currently only a few parameters for each VELO half
  - Should be stored (by definition) as Online Conditions
  - If required also as Offline Conditions, should be replicated in the ConditionsDB in a different folder
  - This ensures reproducibility of trigger results at any point in time
- Online full alignment is not considered as an option
  - Alignment must be stable (including VELO in fact!) to better than the detector precision
  - If alignment changes, it justifies a change of run and probably would occur at a change of fill (no emergency)

Magnetic field condition

- It would be inefficient to consider the magnetic field map as a condition
  - To be reloaded at each fill
  - flipping when analyzing offline events sets
- Proposal
  - Have two maps that may coexist in memory (loaded on demand, but staying)
  - Field polarity is an Online Condition
  - Depending on the polarity, the field service will use the corresponding map.
    - Field service registers for polarity to the Update Manager…

Where are ConditionDB(s)?

- Two logical sub-databases
  - One for Online Conditions: updated in real-time by the Online system.
    - Physically at pit 8
    - Who ensures stability of service? (same question for ConfigDB and PVSS archive DB)
  - One for Offline Conditions: updated offline after careful checking
    - Physically at Tier-0, run by IT
- Both replicated at either site, but only one update site
- Synchronization
  - Use Oracle streaming to populate Offline Conditions in the Online DB and vice-versa

Conclusions

- The proposed scheme fulfills so far all requirements:
  - Usage of CondDB for L1 & HLT defined
  - Full reproducibility of Conditions used Online when running later Offline
  - Simplicity of implementation, using COOL IoV mechanism
  - Single update site for each folder
- A few open questions remain for configuring offline applications from Online configurations depending on time.
  - Needs a careful evaluation of the use case…
- Still to be done:
  - Full definition of Online CondDB services, update mechanism (using DIM/Gaucho)
  - Online updates mechanism (see Clara's talk)