LHCb Conditions Database

Marco Clemencic

marco.clemencic@cern.ch
Overview

- Introduction
  - Definition of “Condition”
  - Requirements
- Conditions Database (CondDB)
- Update Mechanism
- Online Usage
- Deployment
- Summary
Introduction
Conditions

Time-varying non-event data

3 degrees of freedom:
- source
- time
- version

2 categories of conditions:
- off-line cond. (multi version)
- on-line cond. (single version)

LCG is developing a library to handle conditions: COOL (see A. Valassi’s talk)
Requirements

An infrastructure is needed:

- Integrated in LHCb framework (Gaudi)
- Flexible \(\rightarrow\) *freedom for the users*
- Efficient \(\rightarrow\) *reduced overhead*
- Easy to use
Conditions Database
Gaudi Architecture

Natural place for conditions
Detector Data Service

- Contains the “Detector Description”
  - classes providing detector informations
    (main consumers of conditions)
- Objects’ lifetime not depending on event loop
  - they can be valid for a set of events
- XML files for persistency
  - good compromise between human-readable and machine-readable
CondDB Conversion Service

Access point to persistency

CondDB Cnv Svc

Converter

COOL

RDBMS

- Oracle
- MySQL
- SQLite

CondDB Access Svc

Transient Detector Store

Persistency Service

Data Files

Detec. Data Service

Algorithm
CondDB Conversion Service

Access point to persistency

Transient object creation

Algorithm

Detec. Data Service

Transient Detector

Persistency Service

Data Files

CondDB Cnv Svc

Converter

CondDB Access Svc

COOL

RDBMS

- Oracle
- MySQL
- SQLite
CondDB Conversion Service

Access point to persistency

Transient object creation

CondDB Cnv Svc

Converter

COOL

RDBMS

- Oracle
- MySQL
- SQLite

Simplify
Caching
Layers
Currently we do not have any real data to put into the database

To estimate the performances we create a copy of all the detector description in a DB

Loading all the data takes (on a 2.8 GHz Xeon)
- \(~15\) s from files
- 1–2 min. from Oracle server

There is still a lot of room for improvements
Update Mechanism
Object A depends on condition C
The Dependencies

- Object A depends on condition C
  - when C changes, A does something
    (caching data, digest...)

A \rightarrow A.f() \rightarrow C
The Dependencies

- Object A depends on condition C
  - when C changes, A does something
    (caching data, digest...)

- Flexibility
  - more actions
Object A depends on condition C

- when C changes, A does something (caching data, digest...)

Flexibility

- more actions
- more conditions
The Dependencies

- Object A depends on condition C
  - when C changes, A does something
    (caching data, digest...)

- Flexibility
  - more actions
  - more conditions
  - more consumers
The Dependencies

- Object A depends on condition C
  - when C changes, A does something
    (caching data, digest...)

- Flexibility
  - more actions
  - more conditions
  - more consumers
  - more levels
Object A depends on condition C
- when C changes, A does something (caching data, digest...)

Flexibility
- more actions
- more conditions
- more consumers
- more levels

Dynamic
Object A depends on condition C

- when C changes, A does something (caching data, digest...)

Flexibility

- more actions
- more conditions
- more consumers
- more levels

Dynamic

Network of Dependencies
Update Manager Service

- Specialized service dedicated to handle dependencies and updates
- At the begin of each event:
  - find objects needing an update
  - update objects
  - call user functions
Given a network of dependencies:
Update Policy

Given a network of dependencies:

- start from the head (objects without consumers)
Given a network of dependencies:

- start from the **head** (objects without consumers)
- intersection of validities
Given a network of dependencies:

- start from the **head** (objects without consumers)
- intersection of validities
  - **object level** (methods)
Update Policy

Given a network of dependencies:

- start from the head (objects without consumers)
- intersection of validities
  - object level (methods)
  - method level (child objects)
Update Policy

Given a network of dependencies:

- start from the head (objects without consumers)
- intersection of validities
  - object level (methods)
  - method level (child objects)
- invalid object
On-Line
On-Line environment

- Local Area Network in the Pit
- \(\sim 4000\) processes
  - conditions loaded at initialization
    (alignments, trigger configurations, ...)
On-Line environment

- Local Area Network in the Pit
- \(\sim 4000\) processes
  - conditions loaded at initialization
    (alignments, trigger configurations, . . . )

\[\Rightarrow\] impossible to use directly a database server
On-Line environment

- Local Area Network in the Pit
- ~4000 processes
  - conditions loaded at initialization
    (alignments, trigger configurations,...)

⇒ impossible to use directly a database server

we can use the cache of the CondDB Access Service
CondDB and On-Line: Initialization

On-Line Process

DB Reader

COOL

RDBMS

Gaudi
On-Line
Service

CondDB
Access Svc

Update
Mgr Svc

Cache
(in memory DB)
CondDB and On-Line: Initialization

On-Line Process

- DB Reader
  - COOL
  - RDBMS

- Gaudi On-Line Service
  - CondDB Access Svc
  - Update Mgr Svc
  - Cache (in memory DB)
CondDB and On-Line: Initialization

On-Line Process

1. DB Reader
2. COOL
3. RDBMS
4. Gaudi On-Line Service
   - CondDB Access Svc
   - Update Mgr Svc
   - Cache (in memory DB)
CondDB and On-Line: Initialization

On-Line Process

- DB Reader
- COOL
- RDBMS
- Gaudi On-Line Service
- CondDB Access Svc
- Cache (in memory DB)
- Update Mgr Svc

CHEP06, 13-17 February 2006, Mumbai – p. 17/23
CondDB and On-Line: Run

Control System

On-Line Process

Gaudi
On-Line
Service

CondDB
Access Svc

Update
Mgr Svc

Cache
(in memory DB)
CondDB and On-Line: Run

Control System

DB Writer

COOL

RDBMS

On-Line Process

Gaudi On-Line Service

CondDB Access Svc

Update Mgr Svc

Cache
(in memory DB)
CondDB and On-Line: Run

Control System

- DB Writer
  - COOL
  - RDBMS

On-Line Process

- Gaudi On-Line Service
  - CondDB Access Svc
    - Cache
      (in memory DB)
  - Update Mgr Svc
Deployment
Deployment Plan

- Master copies at PIT and CERN (synchronized)
- Copies at Tier-1s
- Expected DB size: few GB
Summary
Summary

- LHCb Conditions Database
  - LGC project COOL for the storage
  - integrated in the Gaudi persistency framework
Summary

- LHCb Conditions Database
  - LGC project COOL for the storage
  - integrated in the Gaudi persistency framework

- Update Mechanism
  - flexible
  - simple to use
  - optimized
Summary

- LHCb Conditions Database
  - LGC project COOL for the storage
  - integrated in the Gaudi persistency framework
- Update Mechanism
  - flexible
  - simple to use
  - optimized
- Online Usage
  - system to publish conditions to the on-line farm