An ODBMS approach to persistency in CMS

V. Innocente[1], *L. Silvestris*[2,1],
For the CMS Collaboration

[1] CERN, Geneva, Switzerland
[2] INFN, Bari, Italy

Abstract

Persistent object management has been always at the centre of the CMS Analysis and Reconstruction Framework (CARF). Already the very first prototypes had an ODBMS database (Objectivity/DB) as the key component.

Today, Objectivity/DB is fully integrated into CARF. The present version of CARF offers persistent object management for event structure and meta-data. In particular it manages raw data from test-beams, simulated objects (particles, hits and digis) and reconstructed objects common to test-beams and simulation.

CARF is used for detector performance and high level trigger studies. A review of experiences will be presented.

Keywords: ODBMS, HEP, LHC, CMS, Persistence, Reconstruction

Introduction

The CMS experiment [1] is one of four approved LHC experiments. Data taking is scheduled to start in 2005 and will last at least ten years. CMS software and computing task [2] will be 10-1000 times bigger than current HEP experiments. Therefore, software should be developed keeping in mind not only performance but also modularity, flexibility, maintainability, quality assurance and documentation: Features not easily achievable using “traditional” HEP software tecnologies and practices.

Since 1995 the CMS Software group has setup several R&D projects [3], [4] in order to test new technologies and new methods which will be used to develop the final CMS software. These R&D activities have identified new software technologies, such as object oriented programming (C++), object data management system (Objectivity/DB), which could be successfully used to develop CMS software. Using these technologies, different prototypes, both for online and offline components, have been developed to validate our choices.

These were successfully tested during 1997 and 1998 test beam periods [5],[6],[8], and today are in production for some of the CMS Test beam areas (X5B,T9,H2b) [7] and in final detector optimisation, high level trigger studies and global detector performance evaluation using ORCA (**O**bject Oriented **R**econstruction for **C**MS **A**nalysis [9]).

CARF

The **C**MS **A**nalysis and **R**econstruction **F**ramework, CARF, is described in detail elsewhere [10]. The framework currently supplies the following functionalities:

- Persistent storage management for both event and non-event data.
- Reconstruction framework: the system that provides the users with the requested reconstructed objects (such as Track, Cluster, Jet...).

- Run-time selection of the Reconstruction Algorithms together with the possibility that an event can contain the same type of objects reconstructed with different algorithms (or different parameters) to allow easy comparison.
- Analysis framework: the system that manages the chain of event filters that constitute an analysis cycle.

Communication with the persistent data store is handled by CARF rather than by any explicit code in the user packages. CARF is used to manage the reconstruction utilising a system of reconstruction on demand in which reconstructed objects are only created/accessed from the store, when they are needed.

Persistency Service

CMS Reconstruction and Analysis Software is required to store and retrieve the results of computing intensive processes (and in general of any process which cannot be *easily* repeated). This responsibility covers also raw data storage as we plan to use the same software framework in both offline reconstruction and online level-three trigger.

Persistent Data

Three major types of information, which require to be made persistent, has been identified:

- **Event Data**: data associated to a given *triggered beam crossing*. They encompass:
  - *Raw Data*: data read directly from the detector and eventually processed online. These are WORM (Write Once, Read Many) data, to be securely stored and never modified;
  - *Reconstructed Data*: data produced by the reconstruction process. They can belong to the whole collaboration, to a physics analysis group or to a single user. (Partial) reprocessing of event data produces new versions of reconstructed data. Reconstructed data may be further structured in several *layers* according to use-cases and access patterns.
  - *Event Tag*: a *small* object which summaries the main feature of an event to be used for fast selection.

  Event data are usually organised in datasets according to run conditions, physics selection criteria and access patterns.
- **Environmental Data**: data describing the state of the environment at the time the event data were produced. They are identified by:
  - *a validity time period* : which allows to relate an event to the corresponding environmental data;
  - *a version*.

  Environmental data encompass:
  - Machine status;
  - Detector status (setup, calibration, alignment, ...);
  - Running conditions;
  - Reconstruction configuration, including algorithm parameters and user options.

  These informations can be produced directly from slow control data acquisition system or from offline processes.

  Due to the fact that environmental data can be updated producing new versions it is required to put in place a mechanism which relates the event data (other than raw) with the environmental data actually used during the reconstruction process.
- **Meta-Data**: data describing other data

  - event catalog
  - statistics.

Persistent Object Management

Persistent object management has been always at the centre of CARF development. It has been always considered one of the major task of the analysis and reconstruction framework. Indeed already the very first prototypes (notably 1997 test-beam prototype) had already an Objectivity database as key component [11].

Today, persistent object management is fully integrated into CARF.

CARF manages, directly or through an utility toolkit:

- multi-threaded database transactions;
- creation of databases and containers;
- meta-data and event collections;
- physical clustering of event objects;
- persistent event structure and its relation with the transient one;
- deferencing persistent objects.

CARF provides also a software middle-layer, mainly in the form of template classes, which helps developers of detector software in providing persistent versions of their own objects.

To avoid transaction overhead event objects are first constructed as transient instances and made persistent (by a copy to their persistent representations) only at the end of the event processing when a decision is finally taken about the fate of the event (send it to oblivion or save it in a particular dataset) and about its classification.

Such a copy is not performed in accessing persistent events: physics modules access directly persistent objects through simple C++ pointers. CARF takes care of all details to make sure that the required objects are actually loaded in memory and that their pointers do not become invalid while the event is processed.

This architecture avoids that detector software developers should become Objectivity experts. Indeed the goal is to make the use of Objectivity completely transparent to physics software developers without making compromises in efficiency. This will also guarantee a painless transition to a new persistent technology if Objectivity/DB prove not to satisfy our requirements.

The present version of CARF offers persistent object management for:

- event structure and meta-data common to test-beams, simulation and reconstruction
- raw data from test-beam;
- simulated particles, simulated tracks, simulated hits and simulated digi;
- reconstructed objects common to test-beams and simulation.

Persistent Event Model

In designing a persistent object model two aspects should be taken into account:

- *The logical model*: which describe persistent capable classes, their relationships and the corresponding navigation paths that a user has to follow (explicitly or implicitly) to obtain a service (in a database application usually access to some information).
- *The physical model*: describing the localisation of the various objects on the storage media. I/O granularity is different than object granularity and the way in which objects are physically clustered together affects considerably the performances of I/O bound applications.

An optimised persistent object model will minimise the number of I/O operations to be performed to satisfy the major use-cases. The critical component of a I/O bound use-case is the *access pattern* i.e. which objects are accessed and in which order. Therefore the design of a persistent object model will require an optimisation of the major access patterns. If a single object model can not optimise all major access pattern the use of techniques such as partial replication and re-clustering could be required.

In a multi-user distributed environment concurrency issues should also be considered. Top-level entry point objects could be required to be accessed at the same time by many applications. These objects are usually also the one which requires to be more often updated when new information is added to the database. These activities could create I/O bottlenecks which could require replication and caching techniques to be solved.

The great majority of HEP applications are data reduction processes: they read a *large* amount of data and produce a summary of it without modifying the input data. Therefore the major use-case is a typical analysis job which access parts of an event and produces high level reconstructed objects, tags or just statistics objects (Histograms..). It should be noted that, depending on the stage at which the analysis is performed a different portion of the event is required to be accessed:

- *raw data* belonging to a single detector in a calibration job (and the successive reconstruction jobs);
- a large portion of the *reconstructed objects* in first pass physics analyses which will produce high-level (global) reconstructed objects;
- *high-level reconstructed objects* and a small portion of the reconstructed objects in final physics analyses.

Other use-cases such as single event visualisation, detailed detector performance studies, should also be taken into account but they can fit a large spectrum of persistent event models without major additional performance penalties.

In the following description of the CMS persistent event model we will use a nomenclature introduced in BaBar [14] and used also in the MONARC [15] project. These concepts, although useful at this stage to describe our model using a common language, will not necessary correspond to concrete entities in CMS final implementation.

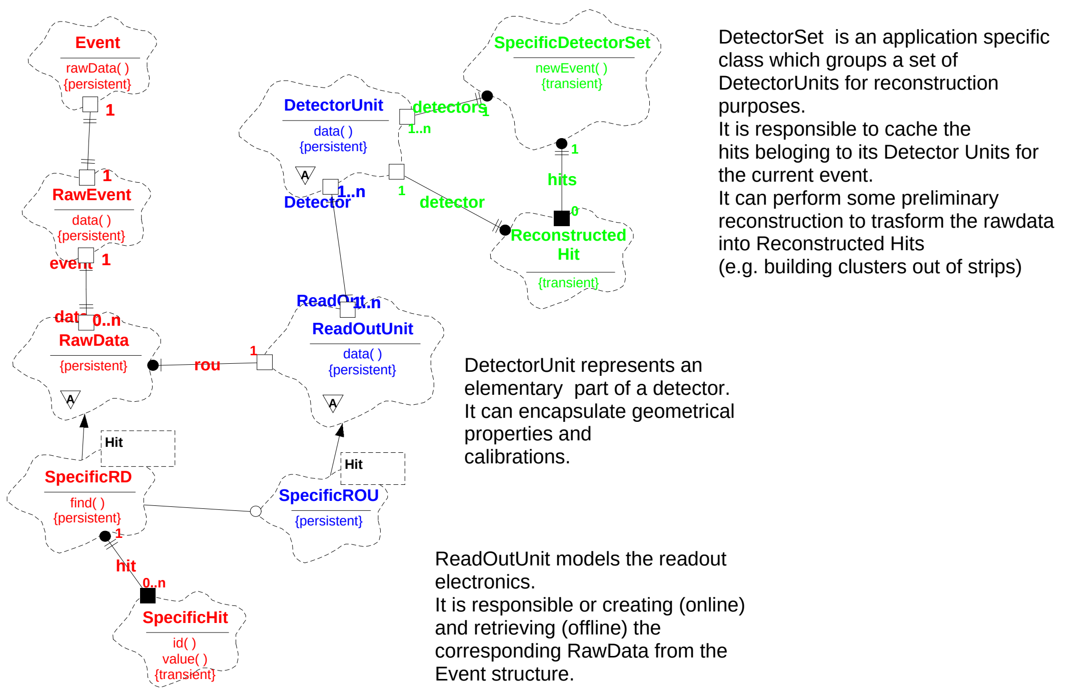
This nomenclature classifies the information belonging to a HEP experiment event (from higher levels to lower levels) according to creation and access patterns in:

- Tag (*Event Selection Tag*): used for fast event selection;
- Aod (*Analysis Object Data*): information used in final analysis;
- Esd (*Event Summary Data*): information required for detailed analysis and high-level (global) reconstruction
- Rec (*Reconstructed Data*): detailed information about reconstructed objects required mainly for detector and algorithm performance studies
- Raw (*Raw Data*): written once, never modified. For simulated events they include also all information generated during the simulation step.

Tag, Aod and partially Esd will be accessed by the majority of the analysis jobs most of which will not be computational intensive. Rec and Raw will instead seldom accessed, mostly by scheduled, computational intensive, production jobs will run over them in a sequential fashion to generate a new version of the corresponding Esd, Aod and Tag. It is interesting to note that analysis jobs which require access to low level information usually require access also to the corresponding high level information and indeed in most of the cases they start from it.

Logical Model

The logical relations reconstructed data and the event as a whole together with the navigation paths used to access them have been already described in detail elsewhere [13].



DetectorSet is an application specific class which groups a set of DetectorUnits for reconstruction purposes.
It is responsible to cache the hits beloging to its Detector Units for the current event.
It can perform some preliminary reconstruction to trasform the rawdata into Reconstructed Hits
(e.g. building clusters out of strips)

DetectorUnit represents an elementary part of a detector.
It can encapsulate geometrical properties and calibrations.

ReadOutUnit models the readout electronics.
It is responsible or creating (online) and retrieving (offline) the corresponding RawData from the Event structure.

Figure 1: Class diagram showing the classes collaborating in raw data handling and detector reconstruction: in the first column are classes whose persistent instances are created event by event, in the second columns are classes whose persistent instances exists for each "set-up", the third column includes transient classes which are application specific

Figure 1 shows the object model that is presently implemented in CARF for raw data access and detector reconstruction.

- **DetectorSet**: represents an application specific transient class which can group several DetectorUnits. It acts as interface among the physics modules, such as pattern recognition algorithms, and the Detector Units. It can cache the "Reconstructed Hits" to enhance performances.
- **DetectorUnit**: models an elementary component of a detector. it can be a persistent class and its instances should be grouped into a "Set-Up" which is managed as a configuration item. It shares with the DetectorSet the responsibilities of caching simulated hits, performing digitisation simulation and detector reconstruction.
- **ReadOutUnit**: models the read-out electronics of a detector. Its major responsibility is to create raw data (in an online DAQ application or during digitisation simulation) and to read them back in offline reconstruction applications. It is a persistent class and its instances are grouped into a "Set-Up" which is managed as a configuration item. One-to-one, one-to-many and many-to-one relations are allowed between DetectorUnits and ReadOutUnits.
- **RawEvent**: The *entry point* to all information about a "raw" event
- **RawData**; A simple persistent collection of the raw data belonging to a given ReadOutUnit for a given event.

Physical Clustering

Raw data belonging to the same *sub-detector* are clustered together. This optimises detector studies and low-level reconstruction use-cases which usually access one type of sub-detector. Raw data

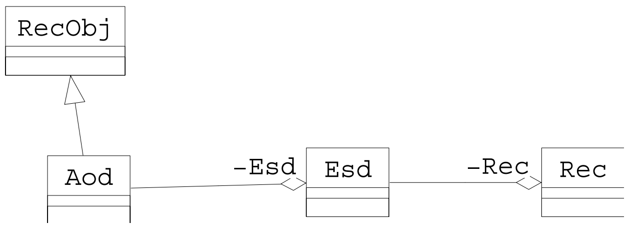


Figure 2: Class diagram representing a possible decomposition of a reconstructed object into Aod, Esd and Rec components

are not required to be modified and no versioning mechanism is provided. This decision could be revised if a re-analysis of simulation use-cases shows the needs of multiple versions of digitisation simulation. Each CMS reconstructed class can be subdivided into Aod, Esd and Rec components according to access patterns. Aod, Esd and Rec will be physically clustered independently. For a given class, such as track or cluster, the Aod will include physics information such as a Lorentz vector and particle identification, the Esd, detector related information such as position or energy deposited in various calorimeter compartments and quality criteria, while the Rec could contain detailed information about the original object constituents such as hits. This subdivision is not rigid: some high-level objects can have only an Aod component while some low-level objects can be completely clustered into Esd containers.

Figure 2 shows a model of a possible decomposition. In this case the Aod is directly inheriting from RecObj (abstract reconstructed-object class) and it is therefore the “real” reconstructed objects. It has as private component a reference to the corresponding Esd which in turn as a the Rec as private component. In this model it is supposed that the Aod explicitly exports (acting as a *proxy*) the Esd services which in turn exports the Rec services. In this way two goals are achieved:

- the user has a unique coherent view of a reconstructed object through the Aod interface, independently of the details of the Aod-Esd-Rec implementation.
- The information can be properly clustered according to the access pattern (moving some information from one component to another require schema evolution)

When re-reconstruction is required (due to changes in calibrations or algorithms) a new version of the reconstructed event (RecEvent) is created. Although the possibility of in-place replacement is still being investigated, all use-cases show that versioning offers more flexibility and safety. The new RecEvent refers to the previous version of all reconstructed objects which do not require an update and holds all the ones which have been re-reconstructed. It should be noted that the model of figure 2 allows the creation of new versions of Aod and Esd without any modification or copy of the lower level information. Indeed different Aod versions can share the same version of an Esd. This allows also an easy partial event replication (just a new version with identical copy of the parts to replicate instead than a real re-reconstruction). In this schema the responsibility of holding the information about which is the *correct* event collection to access is demanded to some meta-data (event catalog). At present CARF manages meta-data mainly through Objectivity naming features.

TestBeam Production in 1999

Since 1997 the CMS software group is working towards a more coherent common online and offline reconstruction and analysis environment, based on common data formats, utility programs and simulation frameworks.

During the prototype phase (1997-1998) a complete processing chain for test beam data (data acquisition, reconstruction and analysis and analysis tools) was successfully tested in three different CMS Test beam areas (H2, T9 and X5b). The data (about 300 GB) were collected using a new data acquisition system in parasitic mode in order to avoid problems to those users that were making detector performance studies.

From the beginning of 1999 the new processing chain is in production for the Tracker and Muon Test Beams [12]. For each test beam, the CMS Test Beam DAQ system stores the data into a separate federated database.

Two federations are used per test beam: one online and one offline. Data are written locally on the online system and then, using an Objectivity version of the Central Data Recording (CDR) package [16], the database files are moved from the disks connected to the online computer system, to the disks connected to the offline data-server and attached to the offline federation (see figure 3). All databases are also stored under HPSS [17].

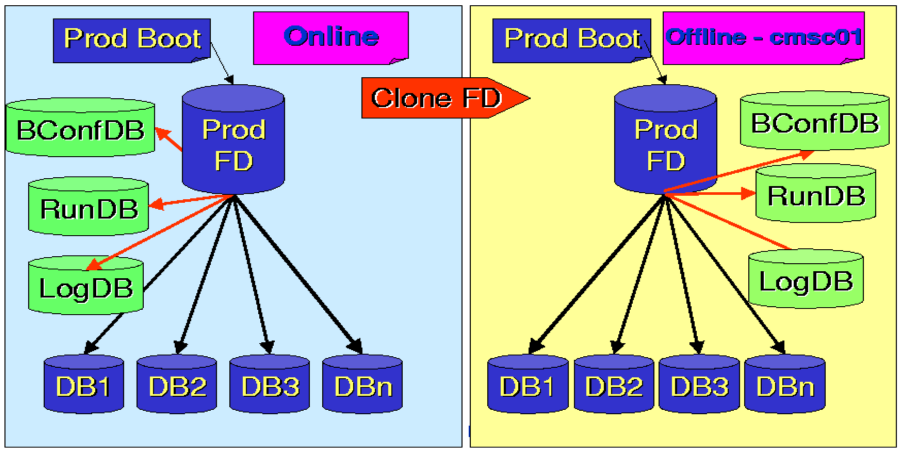


Figure 3: CMS Online and Offline Federations

The total data volume handled by the different test beams during 1999, using this implementation of the CMS Test Beam DAQ system, was 800 GB. It is worth noticing that during the Tracker test beams (X5B, T9) up to 25 concurrent users were accessing data (mostly interactively) on the offline system without any observable degradation of its response.

ORCA Production in 1999

Higher Level Trigger (HLT) studies require persistency to be fully implemented already now. The processing time for high-luminosity ($10^{34} cm^{-2} s^{-1}$) events is too costly to be repeated many times. (For example the ECAL pile-up simulation requires the reading of some 200 minimum bias events, about 70 MB, for every trigger event).

We have built a production system in which we used 19 high-power CPUS (10 Linux Intel-PC PIII 450-MHz, 7 Sun 300MHz UltraSPARC-II and 2 Sun 400MHz UltraSPARC-II) working in parallel and writing into two Objectivity Federations. We were able to process about 30k events/day in this manner. The setting up of these systems had already given us valuable insight into the sorts of problems that can arise in such a complex (for now) environment.

More than 600k events, with pile-up equivalent to LHC operation at a luminosity of $10^{34} cm^{-2} s^{-1}$ were processed and stored in two Objectivity/DB federations. These have been used for Trigger studies and this work is still continuing. This and future ORCA production runs will be the basis for the validation of the Higher Level Trigger design and implementation throughout 2000.

Summary

Since 1995 the CMS software group is working on possible solutions to the problem of persistent data management.

The main conclusions are:

- A persistent object model well matches the description of the data for CMS experiment.
- An Object Data Base Management System (ODBMS) responds to our requirements for all types of data and provides a coherent solution to the problem of persistent object management.
- Objectivity/DB has been identified as possible candidate to be used as ODBMS for CMS. In particular Objectivity/DB has been evaluated in several prototypes which successfully stored and retrieved environmental data, Test-Beam data, simulation data, reconstructed data and statistical data such as histograms. Objectivity/DB also successfully passed several benchmark tests: for instance the ability to write at 170MB/s into a federated database [7].

References

1 *CMS - The Compact Muon Solenoid, Technical Proposal* **CERN/LHCC 94-38, LHCC/P1**, CERN 1994.
2 *The Compact Muon Solenoid, Computing Technical Proposal* **CERN/LHCC 96-45**, CERN 1996.
3 *RD45 - A Persistent object manager for HEP* http://wwwinfo.cern.ch/asd/rd45
4 *Libraries for HEP Computing - LHC++* http://wwwinfo.cern.ch/asd/lhc++/index.html
5 *Status Report of the RD45 Project* **CERN/LHCC 97-6**
6 *Status Report of the RD45 Project* **CERN/LHCC 98-11**
7 *Status Report of the RD45 Project* **CERN/LHCC 99-28**
8 *Status Report of the LHC++ Project* **CERN/LHCC 98-11**
9 D. Stickland, *The Design, Implementation and Deployment of a Functional Prototype OO Reconstruction Software for CMS. The ORCA project.* **Chep 2000 Abstract 108**
10 V. Innocente, *CMS Software Architecture: Software framework, services and persistency in high level trigger, reconstruction and analysis* **CMS/IN 1999-034**
11 L. Silvestris, *A Prototype of the CMS Object Oriented Reconstruction and Analysis Framework for the Beam Test Data* **CMS CR 1998/022** presented at CHEP98, August31 - September 4 1998 Chicago, Illinois, USA
12 L. Silvestris, *CMS Test Beam Software: Online, Persistence, Reconstruction and Analysis* **CMS/IN 1999-043**
13 V. Innocente, *CMS Reconstruction and Analysis: an Object Oriented Approach* **Computer Physics Communications 110 (1998) 192-197** presented at CHEP97, Berlin, Germany
14 *BaBar Offline Software Page* http://www.slac.stanford.edu/BFROOT/www/Computing/Offline/index.html
15 *Monarc project* http://www.cern.ch/MONARC
16 *Central Data Recording using Objectivity/DB* http://wwwinfo.cern.ch/pdp/te/cdr/objy.html
17 *HPSS Services at CERN* http://wwwinfo.cern.ch/pdp/ps/hpss/Welcome.html