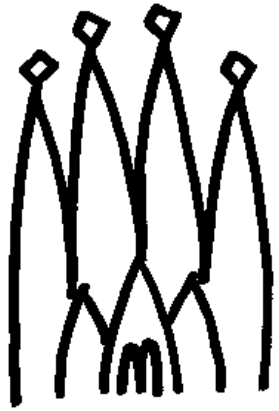
Conditions DB in LHCb

LCG Conditions DB Workshop

8-9 December 2003

P. Mato / CERN



Contents

- Scope of Conditions DB
- Project History
- Requirements
- Integration with Gaudi
- Current Status
- Summary

Conditions DB: Scope

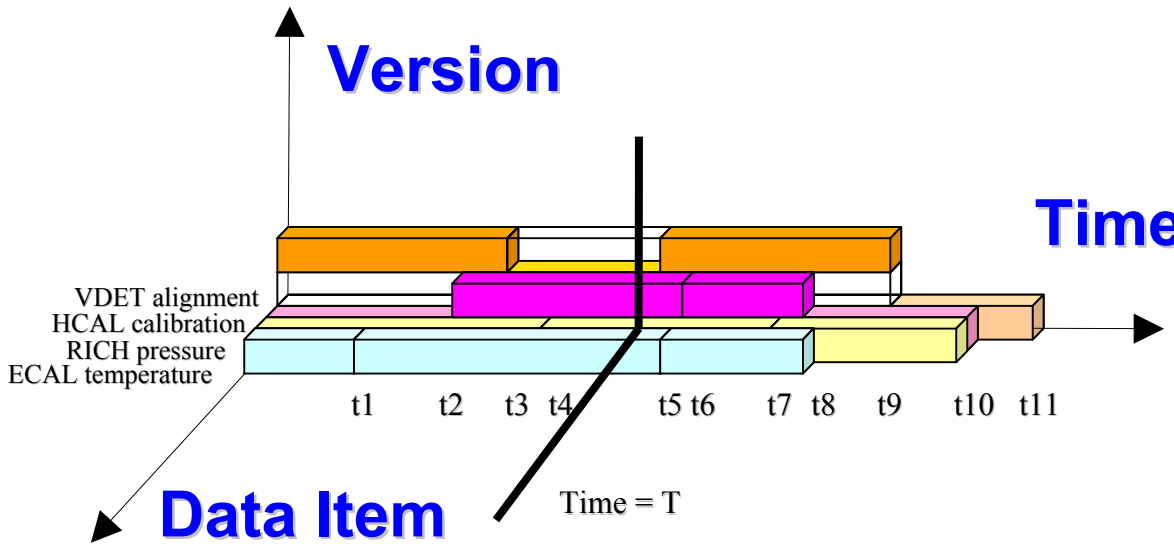
- Everything that is needed in terms of detector conditions by "Offline" Event Data Processing Applications
  - It is not intended for troubleshooting the detector and DAQ system
  - Event Data can not be reconstructed/analyzed without it
  - Time validity period > 5 minutes
- World-wide access
  - Distributed world-wide data analysis
  - GRID-aware (replica, closest available copy, etc.)
- Multi-platform and very long-life time
  - Data analysis will be done in many different platforms
  - Conditions data will need to be available for many years

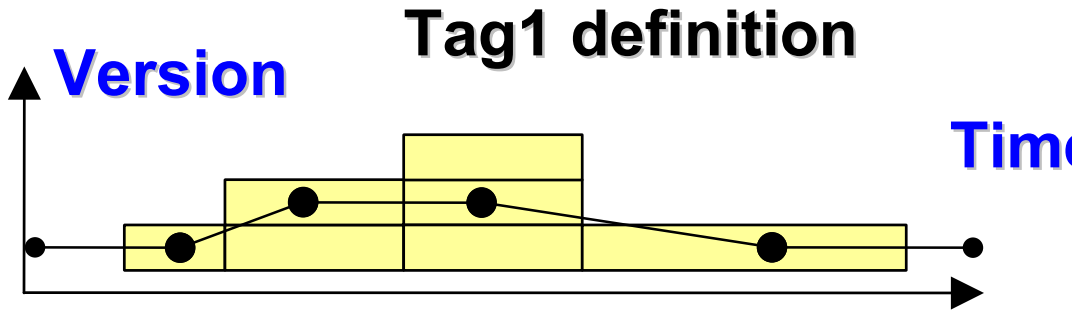
History and Status

- Started several years ago
  - Identified the needs (one of the ingredients of the Gaudi architecture)
  - Defined interfaces in collaboration with other LHC experiments and IT/DB
- Prototypes
  - Developed prototype of integration in Framework
  - Tested Oracle implementation
- Hit by manpower limitations
  - No work done for more than one year
  - Identified as one of the major critical items in recent Manpower Review
  - In the processes of identifying new resources

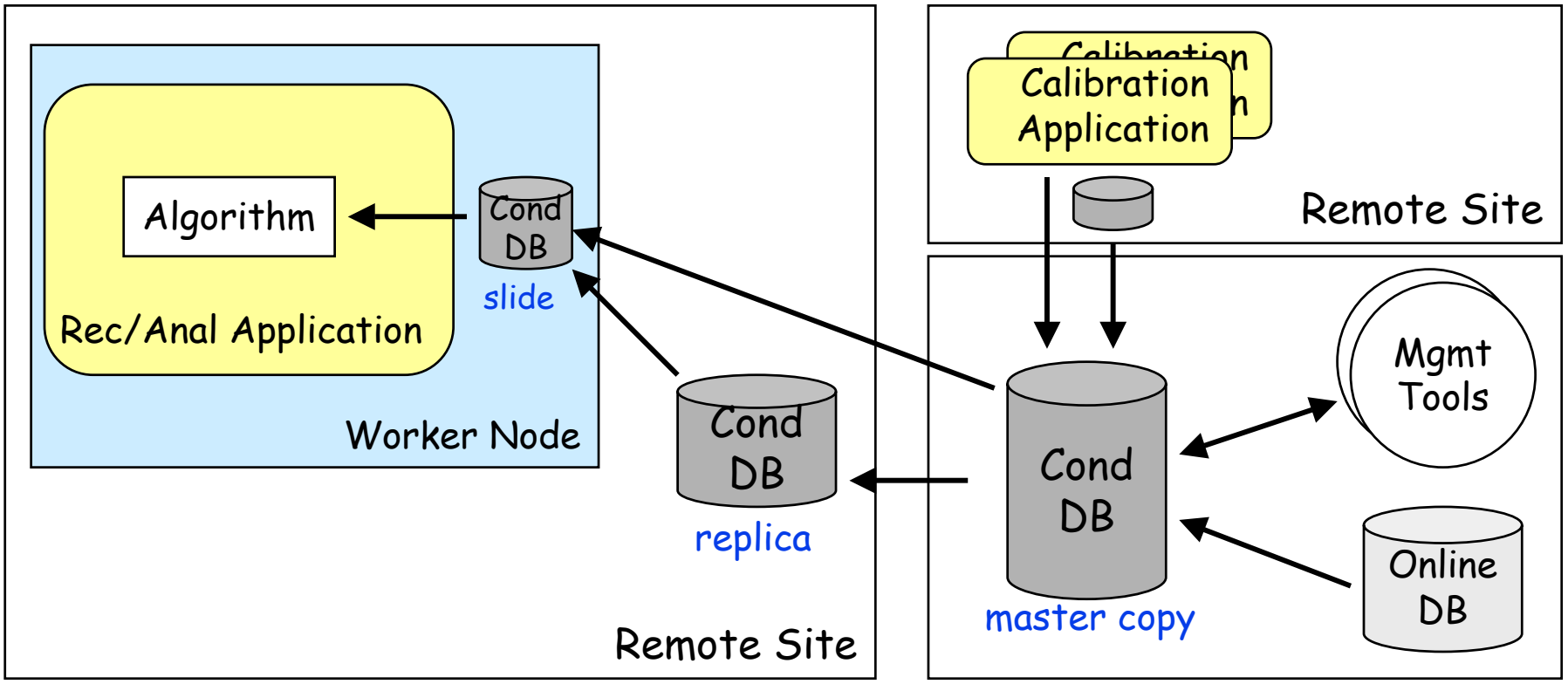
Condition Database Requirements

- Storage/Retrieval of time depend data items
- Versioning
- Tagging
- Ability to extract slices of data
- Not intrusive and as much as possible transparent for the algorithms





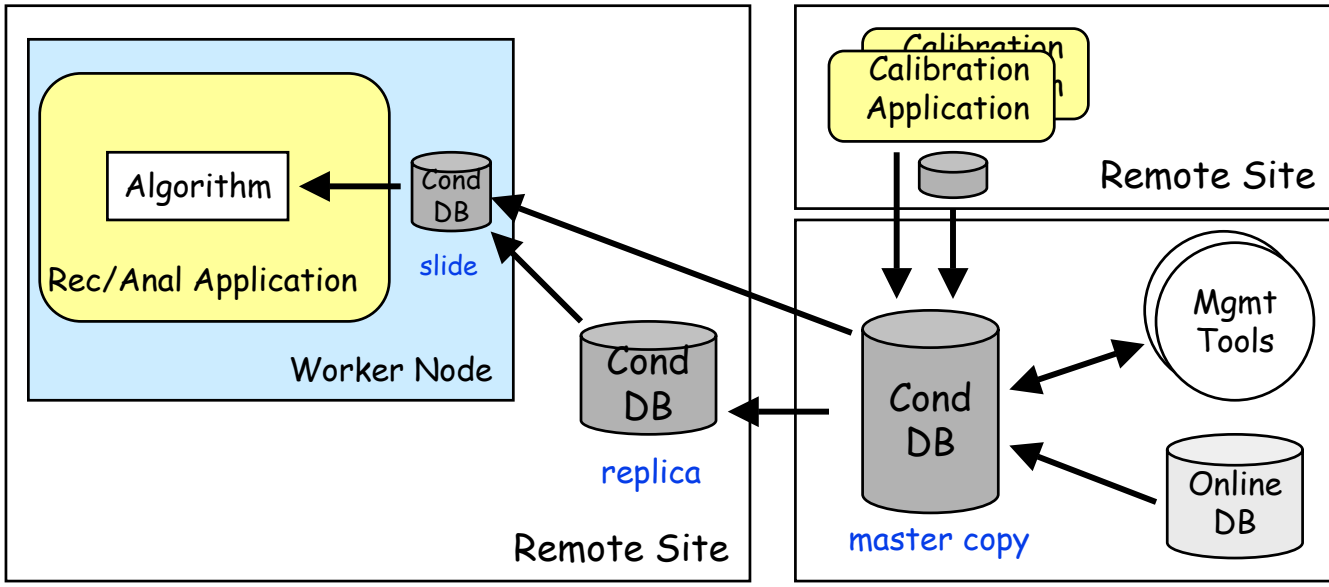
The Big Picture



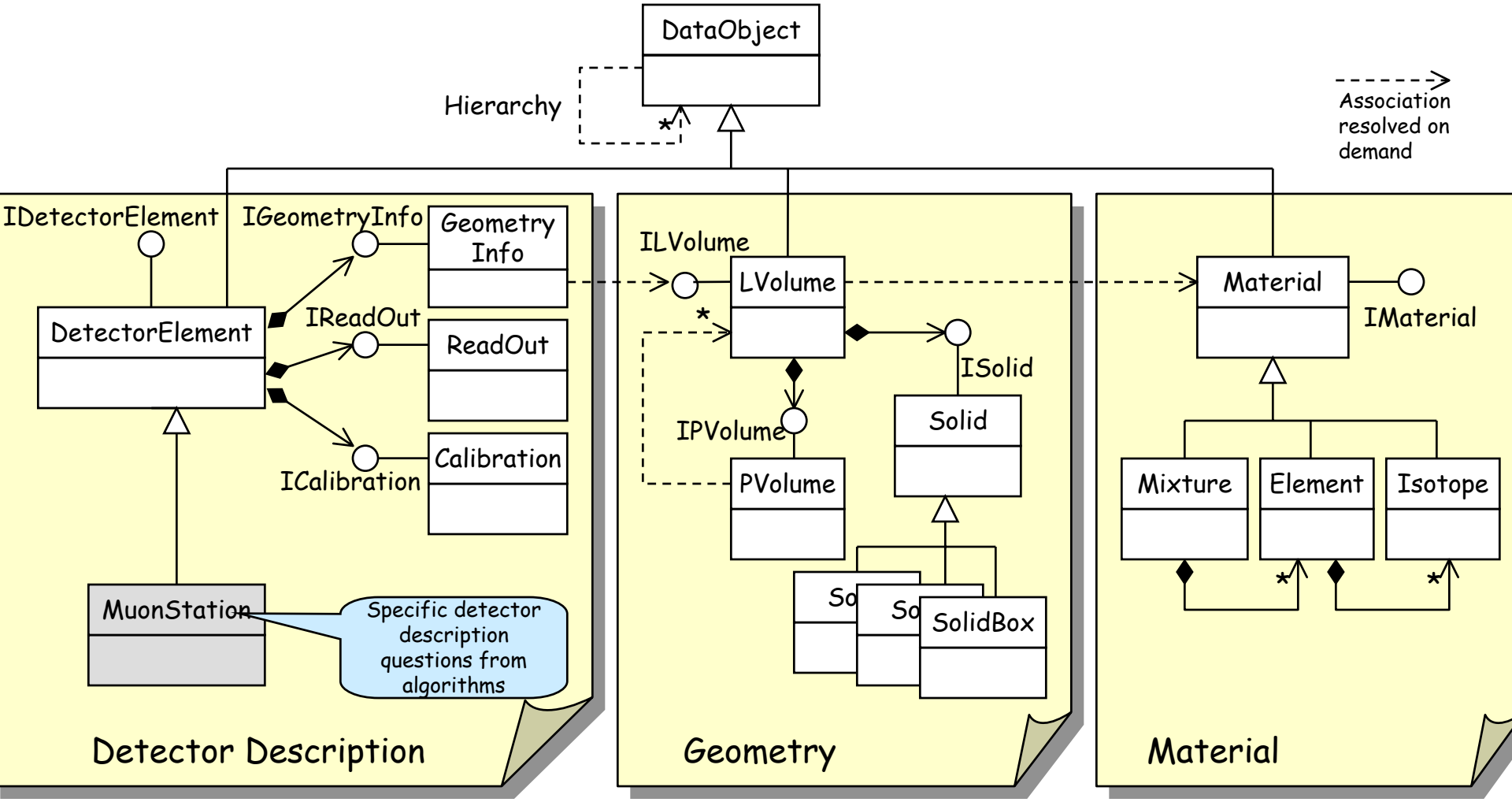
Splitting the Work

- Common:
  - Database implementations (more than one)
  - Tools for management, data browsing, replication, slicing, import/export
  - Deployment of DB services

- LHCb specific:
  - Data contents
  - Interface to Algorithms (Gaudi framework)
  - Data sources: calibration programs, online system, etc.

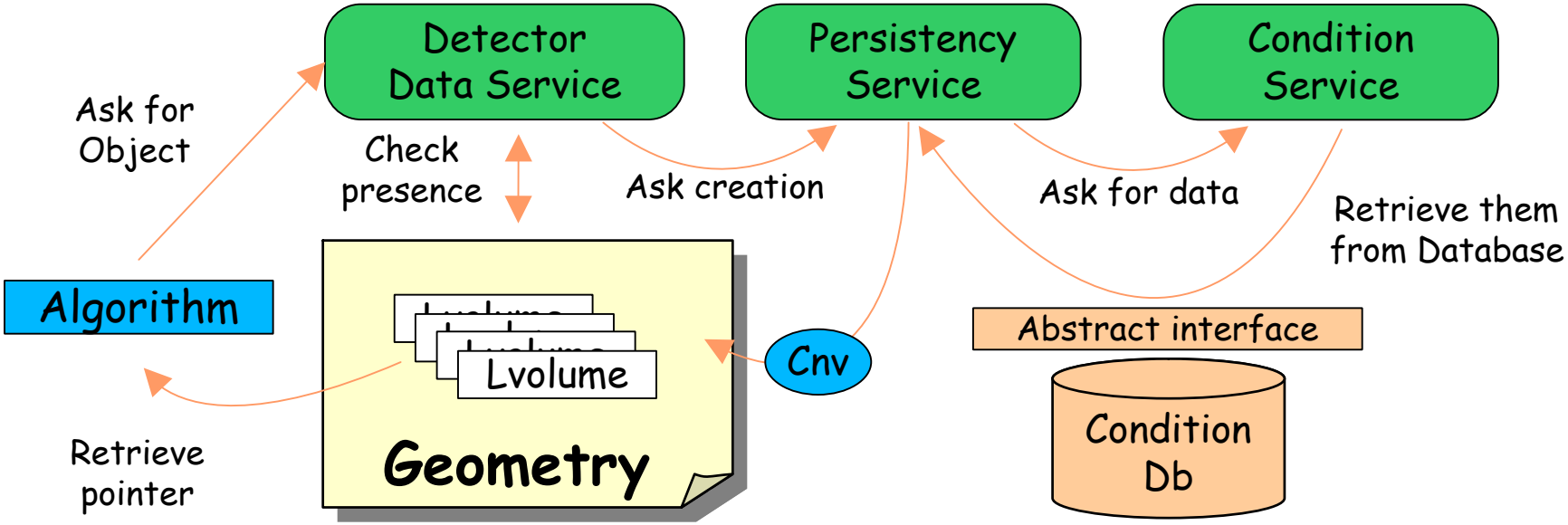


DetDesc Class Diagram (simplified)



Gaudi Interface to Conditions Db

- Emphasis on the data retrieval functionality
- One new service was defined : *ConditionSvc*
- Independent from data content, only deals with data retrieval depending on time, version and/or tag
- Fully transparent for the user



Current Prototype (1)

- We store "verbose" XML in the condition database, exactly as it is in the XML files for geometry
- We use references to point an object in the condition database, with a new protocol called "conddb"

```
<catalog name="Hcal">
  <conditionref href="conddb:/CONDDB/SlowControl/Hcal#scHcal"/>
</catalog>
```

- The retrieval in C++ is absolutely identical, no distinction between condition data and regular data

```
SmartDataPtr<DetectorElement> lhcb(detSvc(), "Structure/LHCb");
if (!sc.isSuccess()) return StatusCode::FAILURE;
Condition* alLHCb = lhcb->alignment()->condition();
```

Current Prototype (2)

- Note the "new" part of the Detector Element API

```
virtual IAlignment*   alignment  ();
virtual ICalibration* calibration();
virtual IReadOut*     readOut    ();
virtual ISlowControl* slowControl();
virtual IFastControl* fastControl();
```

- All these interfaces inherits from IConditionInfo

```
class IConditionInfo ...
  virtual IDetectorElement* detElem() const = 0;
  virtual const std::string conditionName() const = 0;
  virtual Condition* condition() = 0;
```

- Condition is an object which handles a set of parameters, exactly like DetectorElement. It can also be extended in the same ways (including <specific> tag and converters)

Current Prototype (3)

- The way to populate the condition database today is manual
- Look at the examples in the Ex/DetCondExample package

```
StatusCode sc =
  serviceLocator()->service ("ConditionsDBCnvSvc", cdbSvc );
IConditionsDBGate* cdbGate = cdbSvc->conditionsDBGate();
ICondDBMgr* cdbMgr = cdbGate->condDBManager();
ICondDBDataAccess* cdbDatAcc = cdbMgr->getCondDBDataAccess();

cdbMgr->startUpdate();
ICondDBObject* cObj = CondDBObjFactory::createCondDBObject
   (since, till, "**** XML CODE ****", "Comment")
cdbDatAcc->storeCondDBObject
   ("/CONDDB/SlowControl/LHCb/scLHCb", cObj);
CondDBObjFactory::destroyCondDBObject(cObj);
cdbMgr->commit();
```

What we are missing?

- A port of the database to other technologies (MySQL,…)
  - To be able to run in "disconnected mode"
  - Similarly as the POOL file catalogue
- A tool to populate / browse / edit the condition database.
- A tool to make slices of the condition database that you may store locally on your laptop or send to your computing node in a production environment
- Data to try out the current implementation in real conditions
- Develop coherent calibration/alignment procedures among LHCb sub-detectors
- Someone who has time to supervise all this activity

Summary

- Stated long ago to work on Conditions DB but without continuity due to manpower reductions
  - Prototypes to evaluate current ideas exist
  - Not yet used in production
- The LHCb core software team aims
  - To facilitate the task for *Algorithm* writers (framework should take care of most of the work)
  - To encourage common developments between sub-detectors in the areas like alignment, calibration, connection to online system, etc.
- The Conditions DB common project is an opportunity of sharing the work
  - LHCb can not afford to do it alone
  - We have enough work with the LHCb specifics