Data Persistency Solution for LHCb

- Motivation
- Data access
- Generic model
- Experience & Conclusions

M.Frank LHCb/CERN - In behalf of the LHCb GAUDI team
Motivation

➢ Physics software should be independent of the underlying data storage technology
➢ Data of different nature has to be accessed
  ➢ Event data, detector data, statistical data, …
  ➢ The access patterns differ
  ➢ The data set size varies from several Mbytes to 1 Pbyte
  ➢ Legacy data was written in ZEBRA format
➢ It is unclear how these data will be stored
  ➢ Locking into one technology may be a disadvantage
Strategy

- Transient data representation is separated from the persistent data representation
  - Each representation can be optimized separately
  - Transient representation can be used to convert to any other representation
- Minimize coupling between algorithms and the transient data
  - Algorithms see only transient data
  - Transient data items are not intelligent
  - Algorithms post and retrieve transient data from a “black-board”, the data store
How are Data Accessed?

- **Event Data Service**
- **Transient Event Store**
- **Detec. Data Service**
- **Transient Detector Store**
- **Histogram Service**
- **Transient Histogram Store**

**Algorithm**

- Event
  - Anal
    - AxPartCandidates
  - Mc
    - MCParticles
    - MCTrackerHits
    - MCVertices
    - Raw
Functionality of Data Stores

➢ Manage objects of **similar lifetime**
➢ Manage data objects like a librarian
  ➢ Clients store objects
  ➢ Other clients pick up objects when needed
  ➢ Retrieve object collections

➢ Manage **ownership**
  ➢ Does cleanup
Structure of the Data Store

- **Tree** - similar to file system
- **Identification** by logical addresses:
  "/Event/Mc/MCParticles"

- **Tree node**
  - has data members (payload)
  - contains other node objects (directory structure)

- **Browse capability**
Layout of the Data Object

Data Object

/node1
Data Members

/node1/A
/node1/A/A1
/node1/A/A2
/node1/B

Node objects

Symbolic links

/node2
Data Members

/node2/C
/node2/D

GAUDI

M.Frank LHCb/CERN
Store Dynamics: Loading

Algorithm

(1) retrieveObject(…)
Try to access an object data

Data Service

(2) Search in Store

Persistency Service

(3) Request load

Will be unsuccessful, requested object is not present

Request dispatcher Objy, SICB, ROOT,..

(4) Request creation

Converter

(5) Register

Conversion Service

Data Store
Store Dynamics: Storing

Criteria

Output Stream

(1) Collect references

Persistency Service

(2) Create persistent representations

Conversion Service

Converter

Data Store

GAUDI

M.Frank LHCb/CERN
Generic Persistent Model

Transient

Persistent

C++ pointer >> object ID

Objects & pointers

Objects, object IDs, collections & DBs

M.Frank LHCb/CERN
Database Technologies

➢ Identify commonalties and differences

Necessary knowledge when reading/writing

<table>
<thead>
<tr>
<th></th>
<th>Generic</th>
<th>ZEBRA</th>
<th>ROOT</th>
<th>RDBMS</th>
<th>Objy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Write</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>File</td>
<td>File</td>
<td>Database</td>
<td>Database</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td>Bank</td>
<td>Tree/Branch</td>
<td>Table</td>
<td>Container</td>
<td></td>
</tr>
<tr>
<td>Item ID</td>
<td>Record #</td>
<td>Event #</td>
<td>Prim.Key</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>OID</td>
</tr>
<tr>
<td>Database</td>
<td></td>
<td></td>
<td></td>
<td>As for writing</td>
<td></td>
</tr>
<tr>
<td>Collection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

➢ RDBMS: More or less traditional

➢ Objy is different: How to match?
Generic Model: Assumptions

ConversionSvc

Data Store
- Event
- MCEvent
  - DB name
  - Cont.name

Disk Storage
- Object path
- Class ID
- Item ID

Storage type
- DB name
- Cont.name
Generic Model: References

Storage Type
Class ID
Entry ID
Link ID

Link ID  Link Info
...     ...
<number> DB/Cont.name, ...

Technology
Converter

XID

Event
Mc

Lookup table
Generic Model: Extended Object ID

- Storage Type
- Class ID

- Storage Type dependent part

- OID

- OR

- Link ID
- Record ID

Objectivity

ZEBRA
ROOT
RDBMS
Experience & Conclusions

➢ It is possible to write physics data without knowledge of the underlying store technology

➢ Our approach can adopt any technology based on database files, collections and objects within collections
  ➢ ZEBRA, ROOT, Objy and RDBMS
  ➢ We are able to choose technologies according to needs

➢ Overhead of transient-persistent separation looks manageable

Object Evolution

- Handling of dictionary discrepancies between the application and the database
  - “Generic” handling?
    - default values?
  - Architectural problem
    - Handled inside “Converters”
    - Better chance to supply reasonable values