CERN LIBRARIES, GENEVA
SC00001063

8.4.3 User Defined Analysis Attributes

The support for user defined attributes and for efficient selection of data from an ODBMS is based on the tagDB model. The HepODBMS package in LHC++ currently provides two prototype implementations of tag classes, the so-called "GenericTag" and "ConcreteTag". Both implementations share a common interface to the interactive visualisation framework (a set of IRIS Explorer modules) allowing the end user to produce interactively distributions on sub-selected data.
Generic tags are aimed at not too large collections owned by single end users. They provide a simple user interface for creation of tag quantities and eliminate the inconvenience for the end user to formally define a new persistent class when the set of user quantities changes. Generic tags may contain attributes of type *float, double, short, long, char.* Additional attributes may be added after the initial definition. This flexibility makes the generic tag especially useful during the first development phase of an analysis, when the set of quantities used in the analysis tends to change more frequently. Although they carry a slight performance penalty with respect to *concrete* tags, they are more convenient for end users and avoid many of the schema handling issues, described in section 8.8 on page 58.
Concrete tags do have their own schema and are oriented towards large, shared collections, such as collaboration-wide or work group-wide event collections.
Both implementations share a common interface, which is entirely decoupled from the physical storage model. This permits the implementation of different clustering strategies, such as attribute-based clustering - as in column-wise Ntuples - without affecting the user interface.

In the LHC++ model, before one can start to visualise data, one has define and fill a collection of tags, as shown below. In this example, the *generic* tags are used.

```
// create a new tag collection
GenericTag simTag("simulation tag");

// define all attributes of my tags
TagAttribute<long>  evtNo(simTag,"event number");
TagAttribute<float> et   (simTag,"Et particle1");
TagAttribute<float> theta(simTag,"theta particle1");
TagAttribute<short> pid  (simTag,"id particle1");
```

Figure 14 - Creation and Definition of a New Event Tag

These tags are then filled in a typical event loop, as shown below. It is important to note that the tag attributes are handled just like normal C++ variables, adhering to the ODMG philosophy[11].

[11] The programmer should perceive the binding as a single language for expressing both programming and database operations, not two languages with arbitrary boundaries between them [23].

```
while ( evt =  geant->nextEvent() )
{
      simTag.newTag();       // create a new tag

      et    = evt->getPart(1).et;
      theta = evt->getPart(1).theta;
      pid   = evt->getPart(1).pdg_code;
}
```

Figure 15 - Filling a Previously Defined Tag

As has been described above, a fundamental feature of this strategy is the ease in which the full event data can be accessed. This is a significant piece of new functionality that was not possible using PAW+Ntuples.

```
while (atlasTags->next())
{
    if (et > 4.5 &&  sin(theta) > .5) // for selected events…
    { // … fill histograms from the tag…
     cout << "event: " << eventNo << endl;
     etHisto->fill(et);
     thetaHisto->fill(theta);

     // … but also using data from the event.
     nTracks = atlasTags->event->tracking->trackList.size();
     nTracksHisto->fill(nTracks);
  }
}
```

Figure 16 - Accessing the Event Data from the Tag

Having populated a collection of tags - typically, but not necessarily, performed in batch, these data can then be visualised using IRIS Explorer.

As described above, IRIS Explorer is a modular tool-kit. An application can be built visually, or can be predefined, out of the basic building blocks, referred to as *modules*. These modules can be those that are provided with the system, HEP-specific modules, or those from other IRIS Explorer user communities.

A simple application, or *map*, to provide similar functionality to that offered by PAW's Ntuple "Plot" and "Project" commands, requires three separate modules. The first module, a database browser, allows a user to select a previously defined collection of tags. Data are then "passed" to a second module, where further cuts are made, and further data can be derived, or associated data retrieved from the database. Finally, the needed quantities are stored in a histogram, for subsequent visualisation. In fact, the data do not flow from one

module to another - the Objectivity/DB object identifier (OID) is passed between modules, using shared memory or TCP/IP sockets, depending on the nodes on which the modules are run - minimising data copies.

Figure 17 - Interactive Analysis using Explorable Collections of Tags

8.5 Data Analysis - a Physicist's Perceptive

A paper presented at the 1996 HepVIS workshop on data analysis and visualisation in HEP[12] identified many of the key problems of today's systems and attempted to define requirements for a future analysis environment. From a physicist's point of view, the requirements were seen as:

- Correctness,
- Homogeneity,
- Consistency,
- Fault tolerance,
- Ease of use.

Although it cannot be claimed that the use of an ODBMS addresses all of these issues, it can certainly have a major impact on the issue of homogeneity. In today's environment, there are a wide variety of data formats in common use. Even when the underlying system is the same, there can be significant difficulties in accessing the data stored using different packages. ZEBRA alone has both "sequential" (FZ) and "random access" (RZ) formats, although both of these have their own variations (FZ native or exchange file format, binary or ASCII data etc.). In addition, the many packages built on top of ZEBRA (DBL3, FATMEN, HEPDB, HBOOK, OPCAL etc.) have inconsistent interfaces. It is not possible to "make a link" (ZEBRA terminology) between data in say a ZEBRA FZ file, associated calibration constants stored in HEPDB, histograms stored in an HBOOK file and data stored in an Ntuple (even if in the same HBOOK file as the histograms). Attempting to scale such "confusion" to data volumes several orders of magnitude greatly is clearly unlikely to succeed.

On the other hand, an ODBMS permits all of the above data to be stored in a consistent manner, even if physically located in separate containers and/or databases on different servers. The user is exposed to the logical, not physical, view.

[12] See http://www.cern.ch/Physics/Workshops/hepvis/hepvis96/papers.html.

8.6 Experience at ZEUS

Like many existing experiments, the ZEUS collaboration at DESY uses an event directory - in their case based on ADAMO - to speed up event selections. A standard program, called EAZE, is provided to access the event data. Users have to provide 3 user-routines and control cards. Each event has an associated header, which includes the run and event number, and the offset within the mini-DST file to the event. In addition, there are a total of 128 bits for event selection.

Experience has shown that the use of individual bits is somewhat inflexible. As a result, the cuts implied by a given bit tend to be rather loose, and hence many jobs read more events than are required, and perform a tighter cut in their program. The ability to perform selections based upon variables, rather than bits, would clearly help, but at the cost of increased storage.

To test these ideas, ZEUS built a prototype "tag database", based upon Objectivity/DB. The system design goals included:

- It should be easy to add new variables to those already stored in the database,
- It should be simple to change these variables (number, meaning, value),
- The interface between the C++ and Fortran code should be transparent to the users.

The main difference seen by the users is in the control cards used to steer the standard analysis job, EAZE. Examples of steering cards for selection of events from the ADAMO-based event directory and the Objectivity/DB tag database are shown below. As these examples show, the new data card format is more comprehensible. In addition, as the selection of data can be more precise, less data is read, resulting in improved performance. As shown in the table below, a job reading some 2750 events from a total of 45000 runs over 7 times faster using the tagDB implementation.

```
C
ZEUSIO-INFI  /zeus/data/mini95/r011539.z
.
.
ZEUSIO-INFI  /zeus/data/mini95/r012208.z
C
ZeusIO-IOPT DRIVER=IE,ZED
C
ZeusIO-ZCLASS .and. b9
ZeusIO-ZCLASS .and. b10
```

Figure 18 - Example of EAZE Control Cards Without TagDB

```
ZeusIO-INFI ZeusEventStore
ZeusIO-Run ((RunNR=>11539)AND(RunNr=<12208))
C
ZeusIO-IOPT DRIVER=OBJY
C
ZeusIO-Variable (
ZeusIO-Variable (Ee>5) and
ZeusIO-Variable ((Zvtx>-50)and(Zvtx<50))and
ZeusIO-Variable ((Eminpz>35)and(Eminpz<65))and
ZeusIO-Variable (Yjb>0.04)
ZeusIO-Variable )
```

Figure 19 - Example of Control Cards Using TagDB

	User Time (seconds)	System Time (seconds)
ZED (old system)	2756.5	91.7
ZES (new system)	362.7	32.0

Table 4 - Performance Comparison: Run 12075, ET > 30

A number of other tools are provided by the system, include a standard program to generate an Ntuple. This can be run as shown below.

- zessel -f sel.txt -n test.rz -v # Produce an Ntuple in the file "test.rz" using the control cards in the file "sel.txt"

```
ZeusIO-Run (RunNr=11543)
C
ZeusIO-Bit (DST27)
C
ZeusIO-Variable (Yjb>0.7)
```

Figure 20 - control cards (sel.txt) for above Ntuple extraction

Currently, all of the data from 1995 are stored in the database, occupying some 14GB of disk space. 92 physics variables are stored per event, and some 10-30% of analyses are performed using the new system. A second phase is currently under study, whereby some 200 variables would be stored per event. This would include data from 1994 until the present, and require some 150-200GB of disk space.

Future plans include storing the physics data itself in Objectivity/DB, rather than just the tags. At this stage, physics analyses directly from C++ would be supported.

8.7 NA48

NA48 is an experiment at the CERN SPS that studies CP violation. It has recently initiated a project with similar goals to that of the ZEUS experiment, described above. In other words, they plan to implement a database using Objectivity/DB to optimise access to physics data. So far, some 20TB of data have been acquired. This will increase to by a further 100TB by the end of the year 2000. Although these data will not be stored in a database, the volume involved dictates that efficient access is required. In particular, it is essential that only the needed data is cached to disk and read into memory. Although the initial proposal calls for a query language slightly different from that employed by ZEUS, and uses bit information to minimise the storage requirements, it is likely that there will be cross-fertilisation between the two projects, and that a common strategy will be evolved.

```
Burst.microCompactBurstMetaInfo bit [5..10] as integer within [-10..10]

(event.microCompareEventData bit [1..8] as integer within [0..100] ) AND
((event.microCompactEventData bit 20=true) OR
(event.microCompactEventData bit 21=true))
```

Figure 21 - Examples of a Possible Query Language for NA48

8.8 Schema Handling Issues

Before an object can be stored in an ODMG-compliant database, its definition or *schema* must be defined. This is done using in the Object Definition Language or, in the case of Objectivity/DB, using Objectivity's DDL, and is shown schematically in the figure below. (For a more detailed description, see [11].)

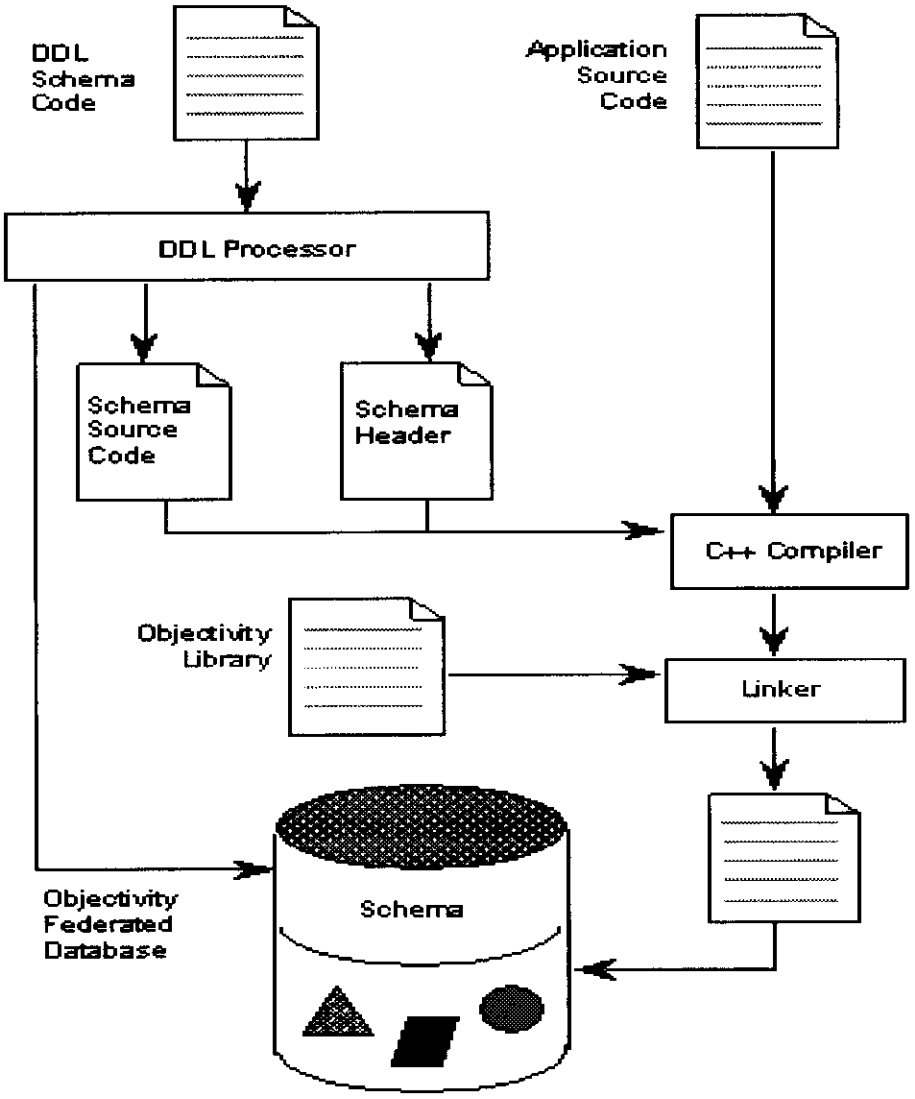


Figure 22 - Database Development Procedure

In the current version of Objectivity/DB, each persistent-capable C++ class is given a type number, which is allocated sequentially. In other words, the type number given to a specific persistent-capable class depends on the order in which the corresponding DDL file is processed. In the current C++ binding this type number is placed as a class variable in the generated code for each persistent class. During the startup phase of an application this number is used to associate the application class with the schema definition for this class stored in the federated database. Maintaining the type-numbering scheme of an application (or library) in agreement with the target federated database schema is therefore an essential requirement to allow the correct functioning of Objectivity/DB.

8.8.1 Schema Consistency between Separated Federations

In a single developer environment, the Objectivity schema pre-processor does guarantee the synchronisation of type numbers, since type number allocation and schema generation is performed against a single federated database. In a larger scale development project with

many software packages and many distributed developers, the constraint to use a single federated database is not practical.
Any schema change performed does require write access to the federated database file for the developer. In order to keep the risk of interference with other users of the federation minimal, we assume that any development will be done against a separate development federation. Only the deployment of stable, released packages should be done against the shared production federation.
To allow developers to set-up private development federations, the development environment must provide a mechanism to create federations containing a copy of the production schema using the same type numbering scheme. This allows the use of production versions of binary libraries of other packages against their development federation. It also simplifies the preparation of input data needed for program testing. One can simply copy test data from the production federation into a database, which is then attached to the development federation.
To simplify the preparation of the development federation schema we have requested a tool, which directly exchanges (parts of) the schema information between two separate federations, without the need to repeat the schema pre-processing step. This functionality will be provided in one of next releases of Objectivity/DB.

8.8.2 Named Schema

To remove the type number coupling between different packages introduced by the sequential type number allocation Objectivity/DB provides the so-called "named schema" feature. This feature allows to divide the type number space of a federated database into named subsets, by specifying the *-schema [name]* when running the DDL processor. Each of these named schemata is reserving a range of 64K type numbers, permitting the individual developer to reorganise the schema within a package without compromising the type numbering of other packages. Some 16 schema names have been allocated for the various LHC++ packages (HepODBMS, HistOOgrams, CLHEP, Geant-4 etc.). The named schema feature has been successfully used to de-couple the development of the different LHC++ packages. We recommend that each experiment register additional named schemata for all experiment specific packages that define persistent classes.

8.8.3 Private User Schema

Requirements for private user schema include the following:

- It should be possible to provide one "named schema", as described above per user.
- Such a "named schema" should be allocated on demand.
- It should support the use of the same class name by multiple users without interference.
- It should be semantically sufficient to implement private end user schema.

Although not a hard requirement, an implementation that kept the user schema in a separate file outside of the central federated database would be preferred, for security reasons.

8.8.4 Dynamic Schema Binding

As discussed above, the static binding of C++ application classes to database schema, using type numbers compiled into binary libraries and applications, has some disadvantages. It complicates the development of federation-independent class-libraries and requires a rather complicated schema preparation procedure if the number of database developers and packages becomes large.

The Java binding to Objectivity/DB provides a more flexible solution. In this case the binding of application classes to the federation schema is done at application runtime using the class name. Using this dynamic binding technique the same application or library can be used against different federations independent of the sequence in which the schema has been defined. We expect that a similar implementation for the C++ binding would greatly simplify the development cycle. A more dynamic schema binding has therefore been requested as a longer-term solution (see section 12.4 on page 94.)

8.8.5 Conclusions on Schema Handling

A strategy for handling both developers' and end-users' schema has been developed and tested. Although received too late for inclusion in this report, new developments in Objectivity/DB for schema exchange and for run-time access to schema appear to largely meet our requirements. Further enhancements in this area are expected from Objectivity/DB and will be discussed at future RD45 workshops.

8.9 Conclusions

The use of an ODBMS as the basis for a consistent, experiment-wide, data management scheme has clear advantages, which have already been demonstrated in production in a number of experiments. These advantages address a number of the requirements, such as homogeneity and ease of use, listed in the CMS Computing Technical Proposal [28]. Further developments of the interactive data analysis environment are clearly required and are already underway. Enhancements to the way that schema are handled, particularly for large projects, where the schema must be shared between multiple federations, have been requested. More prototyping of collections and naming schemes needs to be performed, for which realistic use cases are required. We believe that these activities are best covered as part of the production services that are currently being established.

Additional research needs to be performed in order to understand issues related to the distributed environment of HEP, including data import/export, networking issues, the possible use of technologies such as mobile Java agents, and so forth.

9 Milestone 3

The third milestone set at the March 1997 review of the RD45 project was as follows:

"Demonstrate the feasibility of using an ODBMS and MSS at data rates sufficient for ATLAS and CMS 1997 test-beam requirements."

Since this milestone was set, ATLAS postponed their plans to evaluate Objectivity/DB in a test beam environment until 1998, and hence we only report below on the experience gained in CMS. Furthermore, as the data volumes planned for CMS were of the order of 100GB, it was agreed that these tests would concentrate on the use of Objectivity/DB alone and not address its integration with a mass storage system. Finally, the data rates involved in the CMS test beam activities were rather modest - well below 1MB/second - and hence did not pose any difficulty to Objectivity/DB. Thus, the main challenge posed by the CMS test beam activities was a production demonstration of the overall LHC++ environment, from data taking to analysis - a somewhat different focus to that described in the milestone above.

In addition to the CMS test beam activities, we describe progress on the interface between Objectivity/DB and HPSS, including performance and functionality tests. Plans for test beam activities in 1998 are also included.

9.1 ODBMS - MSS Interface

The need for an interface between the object manager layer and a mass storage system was identified as part of RD45's activities during its first year and is described in [11] and [12]. In summary, although one can expect significant advances in disk capacity/unit price between now and the startup of LHC, it is unlikely that one will be able to afford, or even manage, disk farms capable of storing the entire LHC data volume - a total of some 100PB. More reasonably, one could expect to cache some tens to hundreds of TB of active data on disk, whilst keeping the bulk of the data on cheaper storage media.

Objectivity/DB and HPSS are emerging as the de-facto standard solutions for the HEP community in their respective areas. Plans to use both of these systems in production exist at BNL, CERN and SLAC. As a community, we have requested an interface between these two products, as described in section 6.2 on page 26.

We describe below tests of the prototype version of the interface between Objectivity/DB and HPSS and discuss possible enhancements for the production version of this interface, scheduled for delivery by the end of 1998.

9.2 The Objectivity/DB - HPSS Interface

9.2.1 Introduction to HPSS

The High Performance Storage System (HPSS) is a software system that provides hierarchical storage management and services for very large storage environments. HPSS is the result of a collaborative effort by leading US Government supercomputer laboratories and industry to address very real, very urgent high-end storage requirements. HPSS is offered commercially by IBM Global Government Industry, Houston, Texas and is built upon the IEEE Reference Model for Open Storage Systems Interconnection, more commonly known by its previous name of IEEE MSS Reference Model, shown below.

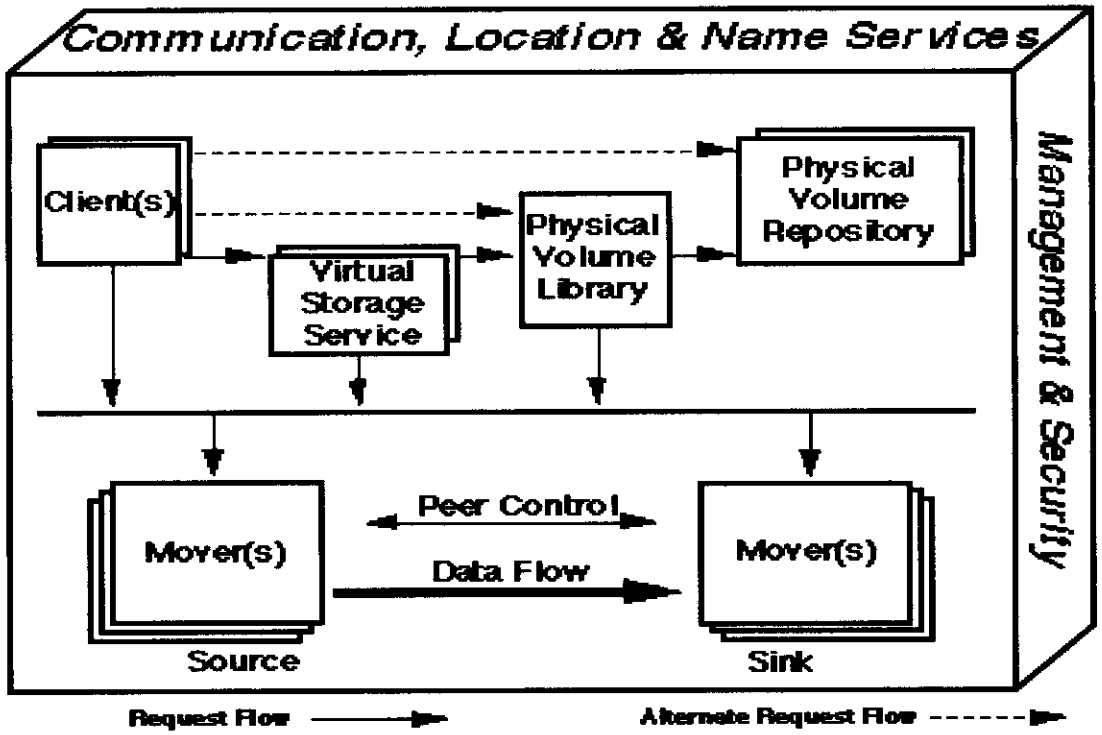


Figure 23 - The IEEE Reference Model for Open Storage Systems Interconnection

HPSS is designed to be scalable in terms of data capacity (up to the level of petabytes), data transfer rates (gigabytes per second), number of files (billions), maximum file size (2^{64} bytes), and geographic distribution of both software components and storage devices.

HPSS achieves these scalability features by supporting both direct- attached and network-attached disk and tape storage devices from multiple vendors, as well as by enabling distributed, parallel I/O through software striping.

HPSS is currently in production at a number of sites, including Maui High Performance Computing Center, Cornell Theory Center, Sandia National Laboratory, Caltech, Fermilab, Lawrence Livermore and Lawrence Berkeley National Laboratories, University of

Washington, Los Alamos National Laboratory, San Diego SuperComputing Center, Oak Ridge National Laboratory, NASA Langley Research Center, Rechenzentrum der Universität Stuttgart, SLAC and CERN.

As such, it has clearly established itself as the mass storage system of choice for sites with high-end requirements. A long list of enhancements are planned, which can be viewed at the HPSS web-site[13].

9.2.2 Control and Data Flow in HPSS

The figure below (page 65) shows the flow of control and data of a read operation for a file stored in HPSS-managed storage. HPSS consists of the following software components:

- Name Server (NS)
  - Maps file name to an HPSS object (bitfile, directory or link)
  - Name Server database is stored by Encina SFS.
- Bitfile Server (BFS)
  - Provides abstraction of logical bitfiles.
  - Supports random access by address and length
- Storage Server (SS)
  - Maps storage segment (SSEG) references into virtual and physical volumes.
  - Schedules mounting.
- Mover (MVR)
  - Transfers data from source device to sink device
- Physical Volume Library (PVL)
  - Manages all HPSS physical volumes.
  - Maps physical volumes to cartridges.
- Physical Volume Repository (PVR)
  - Manages all cartridges

In the following diagram, the first step is performed upon file open. Should the file be disk resident, a read request will execute steps 2, 3, 4 and 7. In the case that the file is offline, steps 5 and 6 are performed, followed by the normal read loop.

[13] http://www.sdsc.edu/hpss/.

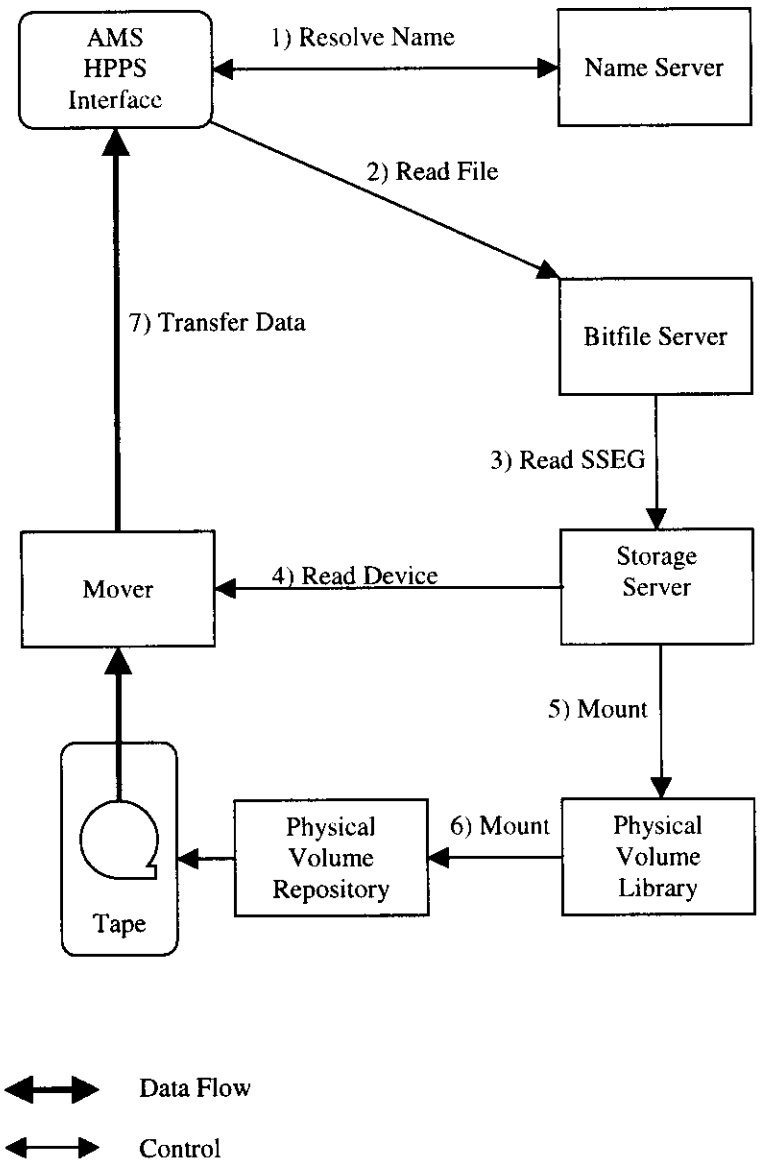


Figure 24 - Control and Data Flow in HPSS

9.2.3 The Objectivity/DB - HPSS Interface

As described in section 6.2 on page 26, an interface between Objectivity/DB and HPSS has been requested by a number of HEP laboratories. A prototype of such an interface has been produced for IBM AIX systems - the only system on which HPSS is currently officially supported. This interface combines the Objectivity/DB server with the HPSS client, and was built by Andy Hanuchevsky/SLAC and Urs Bertschinger/Objectivity. The prototype consists of two parts:

1. A linkable version of the Objectivity/DB server,

2. A library that interfaces the I/O services expected by Objectivity/DB with the equivalent HPSS functionality.

By providing the interface in this way, end-user sites are able to optimise the I/O layer, or even substitute a different mass storage system, provided that a compatible interface is written. Objectivity/DB applications will be unaware that the associated data resides in HPSS managed storage. When an object is accessed, it will be returned immediately if the corresponding database is already disk resident. If not, the client will block on the implicit database open whilst the server, through HPSS, causes the necessary file to be reloaded from tape.

The current interface, which permits one block to be read at a time, is likely to be sub-optimal, but was provided for convenience. A better strategy would be to read multiple blocks at a time, and hence minimise the interaction with the HPSS server. However, the performance implications of the current prototype are not yet well understood, and it is expected that stress testing over the coming months will suggest areas where improvements are required.

Areas where enhancements are expected include:

- Handling of the RPC timeout in Objectivity/DB. The current version of Objectivity/DB has a single timeout value for all RPCs. Although this value can be changed, the default value of 25 seconds is too short to cater for the case when a database resides on tape, particularly if there are limited resources (tape drive contention, stage-out in progress etc.),
- A means of passing "hints" between client and server, such as an indication as to which pages might be read shortly,
- A means of specifying the HPSS "class-of-service" parameter when creating new databases.

9.2.4 The Objectivity/DB - HPSS Installation at CERN

In the current HPSS test configuration at CERN, the various HPSS components are distributed across multiple systems. For example, the tape mover(s), disk mover(s) and HPSS nameserver all run on different systems. In addition, an IBM system is currently being used to evaluate the Objectivity/DB - HPSS prototype interface. As such, this system runs both the Objectivity/DB server (AMS) and the HPSS disk mover, together with the rest of the environment required by HPSS, such as DCE. It is anticipated that one and eventually several/many disk servers will be run for each experiment, each supporting a few hundred GB of disk space managed by HPSS and the Objectivity/DB server.

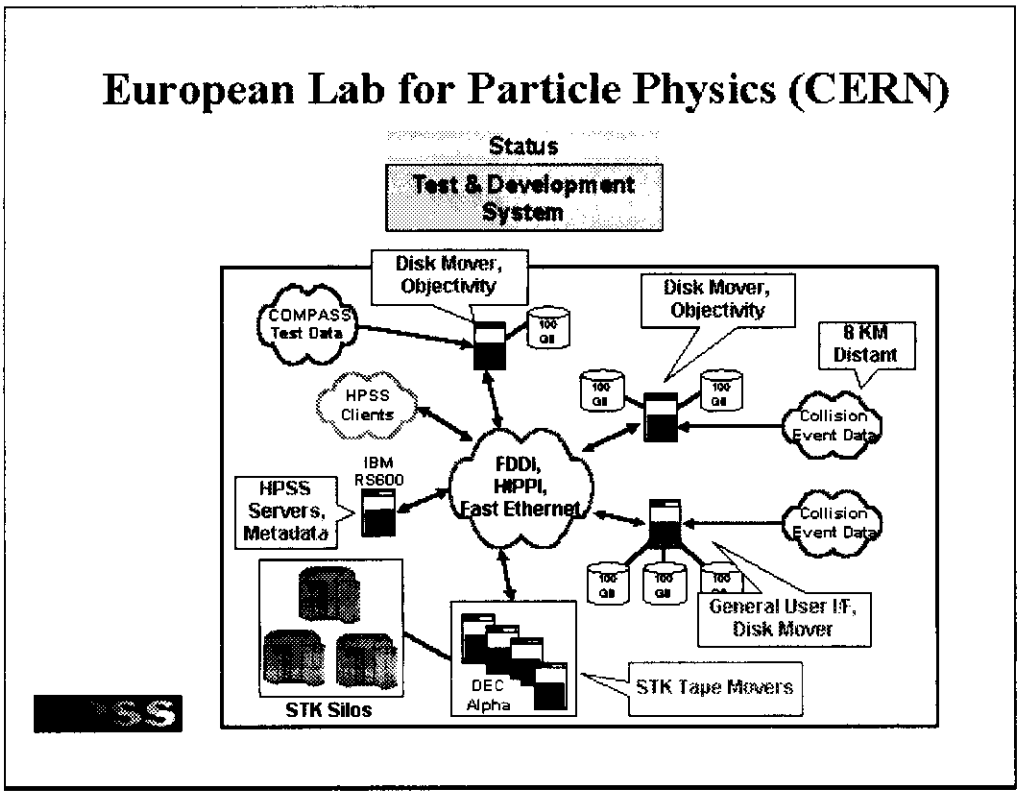


Figure 25 - Objectivity/DB - HPSS Configuration at CERN

9.2.5 The Objectivity/DB - HPSS Configuration at SLAC (BaBar)

Unlike at CERN, SLAC currently plans to run the various HPSS components and the Objectivity/DB server on a single, powerful system. Although such a scenario has the advantage of reducing the network overhead involved in the inter-module communication, it is inherently a less scalable scenario, but nevertheless well-suited to the environment at SLAC, where the system will be used to support a single experiment (BaBar).

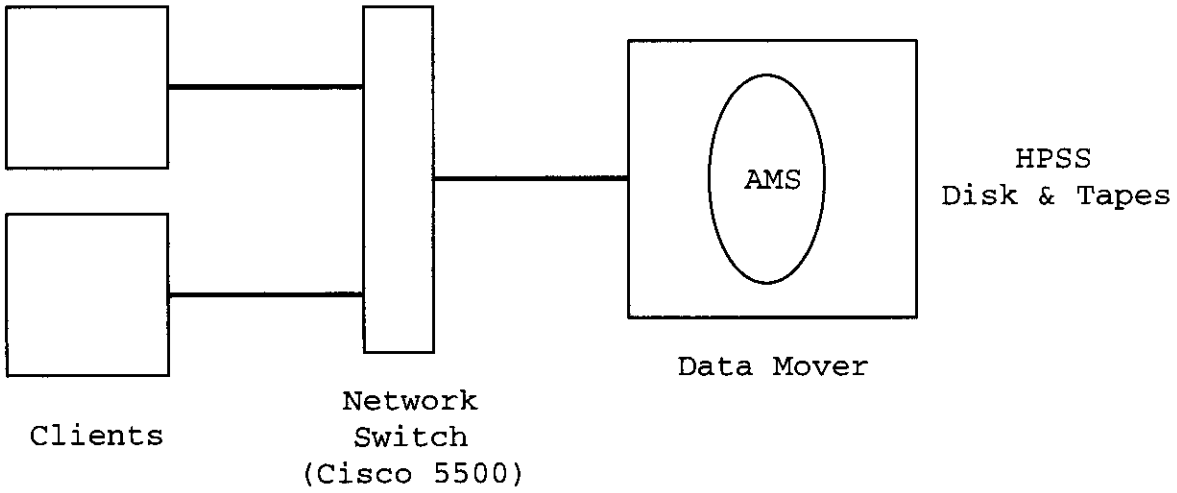


Figure 26 - Objectivity/DB - HPSS Configuration at SLAC

9.2.6 Functionality Tests

The basic functionality required of the proof-of-concept prototype, as described in section 6.2.1 on page 26, have been demonstrated. It should be noted, however, that as HPSS uses DCE security, the Objectivity/DB server has to have the appropriate DCE credentials. As such, the familiar problem of token expiry must be handled.

```
[rshpss01] ~ dce_login # Acquire DCE token
Enter Principal Name: mnowak
Enter Password:
[rshpss01] ~ % cd ~/objectivity/bin # Directory containing modules linked with HPSS API
[rshpss01] ~/objectivity/bin % ./oostartams # Start the modified AMS

Objectivity/DB (TM) Start AMS Utility, Version 4.0.10
Copyright (c) Objectivity, Inc 1989, 1996. All rights reserved.

The AMS has been started (process ID = 52260).
```

Figure 27 - Starting the HPSS version of the Objectivity/DB server

Once the modified version of the Objectivity/DB server has been started, standard Objectivity/DB tools or applications can be used. For example, the *oonewdb* tool is used below to create a new database in the federation "BIG", whose bootfile is also given below.

```
[cernsp] ~/amstest % more BIG
ooFDNumber=1452
ooLFDNumber=65535
ooPageSize=8192
ooLockServerName=rsobjy01
ooFDDBHost=f-rsobjy01
ooFDDBFileName=/objy01/BIG.FDDB
ooJNLHost=rsobjy01
ooJNLPath=/objy01
```

Figure 28 - Bootfile for the "BIG" Federation

```
[cernsp] ~/amstest % oonewdb -db test5 -host f-rshpss01 -filepath . BIG

Objectivity/DB (TM) Create Database Utility, Version 4.0.2
Copyright (c) Objectivity, Inc 1992, 1996. All rights reserved.

Created Database test5 [DBID = 16].
```

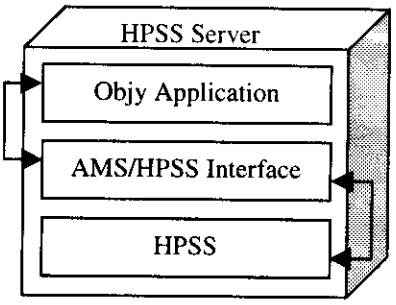
Figure 29 - Creating a New Database in HPSS-managed Storage

```
[cernsp] ~/amstest %oodumpcatalog BIG

Objectivity/DB (TM) List Database Files Utility, Version 4.0.2
Copyright (c) Objectivity, Inc 1990, 1996. All rights reserved.

FD Name   = BIG
FD ID     = 1452
FD File   = f-rsobjy01::/objy01/BIG.FDDB
Boot File = rsobjy01::/objy01/BIG
Jnl Dir   = rsobjy01::/objy01
Lock Host = rsobjy01

...

DB Name   = test5
DB ID     = 16
DB Image  = f-rshpss01::./test5.BIG.DB
```

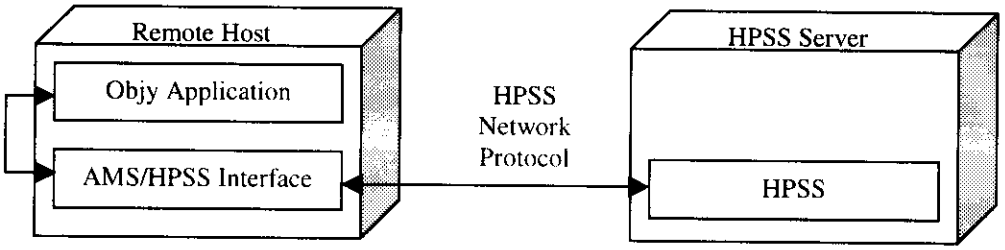
Figure 30 - Output of *oodumpcatalog*

Tests have also been made of access to tape-resident databases. To perform these tests, a federation of two databases was created. Using HPSS administration commands, the two database files were forced to tape. A simple application was then run against the federation. When the application attempted to access the databases in question, they were transparently recalled to disk by HPSS, during which time the application was blocked. As soon as they were disk-resident, the application continued as normal.

1) All modules running on the same host



2) AMS running on a remote host



3) AMS running on HPSS server (or mover)

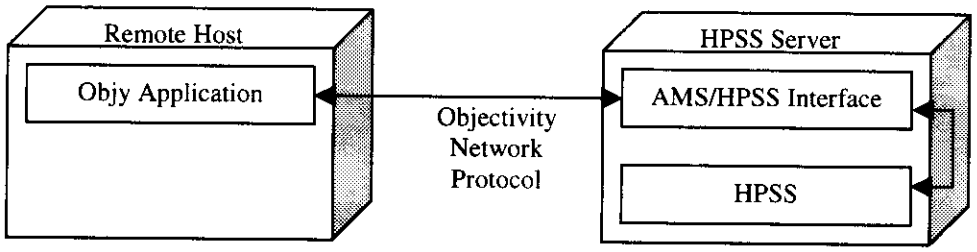


Figure 31 - Tested Objectivity/DB - HPSS Configurations

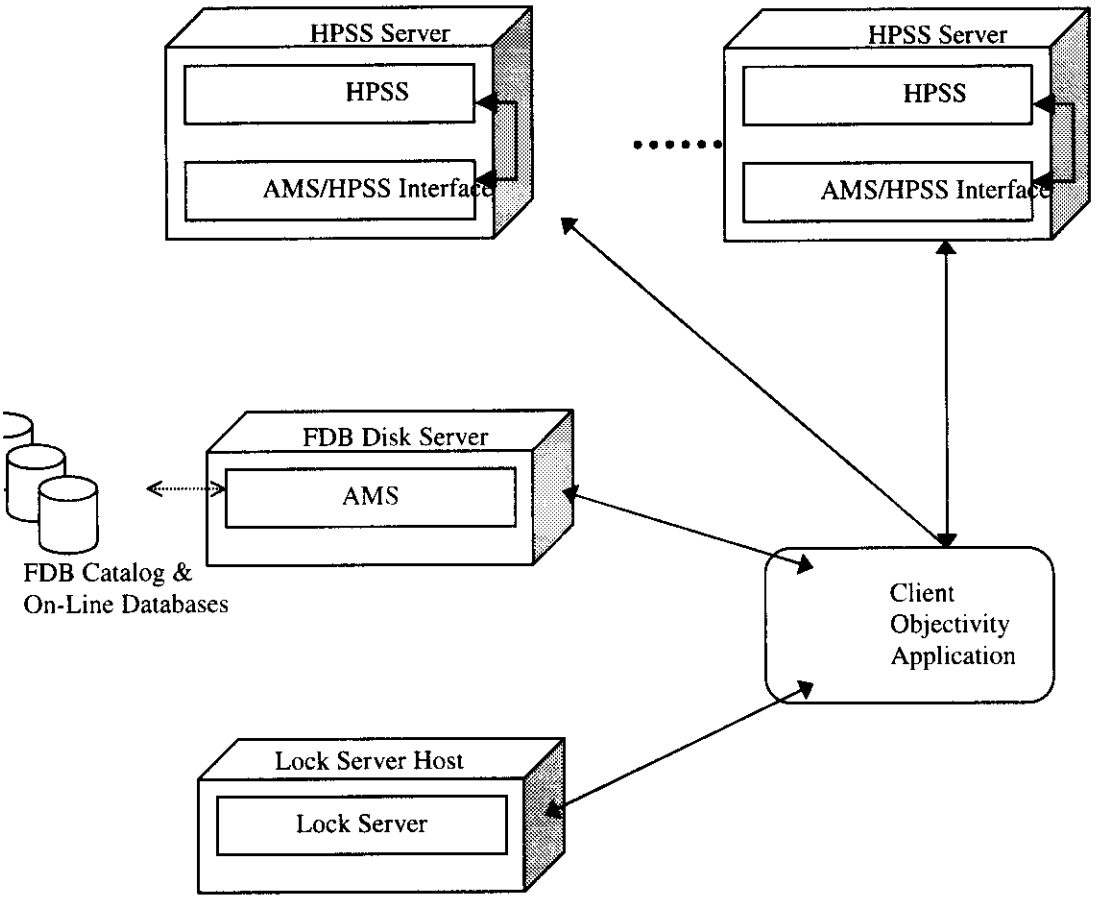


Figure 32 - Objectivity/DB - HPSS Configuration

9.3 Trace of Objectivity/DB I/O Operations

The modified Objectivity/DB server permits us to introduce additional code at the I/O level. For example, this permits an interface to an alternative MSS to be built, provides an exit for access control, and permits I/O operations to be traced.

In the following figure, an application first initialises the federation and opens a database for write access.

```
Start of the transaction

oofs: opening file /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 0 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 0 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 1 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 3 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 5 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 7 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 13 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 0 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 1 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 3 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 59 /user/test.DB...
oofs: Reading 1 pages (8192 bytes, from page 3168 /user/test.DB...
```

Figure 33 - Trace of opening a Database for Write Access

As a second step, the application loops, creating objects of 100KB. Initially, no I/O is performed as the objects are stored in the client cache. Once the cache limit has been reached, data is written to disk. As the objects are large, the I/O is performed in multiple transfers. The first object appears to use a free database page, whereas the second and subsequent objects are written to adjacent pages in a regular pattern. In the case of objects larger than a single database page, the last page contains the page map for the object and is hence written separately.

```
        Cache is full, 3 objects are forced to disk:
               (first updating some internal information)

oofs: Reading 1 pages (8192 bytes), from page 0 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 0 /user/test.DB...
oofs: synching /user/test.DB...

               (writing data)
               first object:

oofs: Writing 8 pages (65536 bytes), from page 3169 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3177 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 2778 /user/test.DB...
oofs: Writing 2 pages (16384 bytes), from page 3178 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3180 /user/test.DB...

               second object:

oofs: Writing 8 pages (65536 bytes), from page 3181 /user/test.DB...
oofs: Writing 4 pages (32768 bytes), from page 3189 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3193 /user/test.DB...

               third object:

oofs: Writing 8 pages (65536 bytes), from page 3194 /user/test.DB...
oofs: Writing 4 pages (32768 bytes), from page 3202 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3206/user/test.DB...
```

Figure 34 - I/O Log

Finally, at transaction commit time, the remaining objects in the client cached are flushed to disk and a *sync* operation performed.

```
Commit of the transaction, flushing the cache:

oofs: Reading 1 pages (8192 bytes), from page 6 /user/test.DB...
oofs: synching /user/test.DB...
oofs: Reading 1 pages (8192 bytes),from page 0 /user/test.DB...
oofs: Writing 1 pages (8192 bytes),from page 0 /user/test.DB...
oofs: synching /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3428 /user/test.DB...
oofs: Writing 8 pages (65536 bytes), from page 3415 /user/test.DB...
oofs: Writing 4 pages (32768 bytes), from page 3423 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 3427 /user/test.DB...
oofs: Writing 8 pages (65536 bytes), from page 3207 /user/test.DB...
oofs: Writing 4 pages (32768 bytes), from page 3215 /user/test.DB...

...

oofs: Writing 1 pages (8192 bytes), from page 3414 /user/test.DB...
oofs: Reading 1 pages (8192 bytes), from page 59 /user/test.DB...
oofs: Writing 1 pages (8192 bytes), from page 59 /user/test.DB...
oofs: synching /user/test.DB...
oofs: closing file /user/test.DB...
```

Figure 35 - Transaction Commit Log

9.3.1 Performance Measurements

In the following figure, we show the performance of HPSS using the so-called "simple API" as a function of blocksize. This API closely resembles the POSIX filesystem interface. In other words, for each POSIX I/O call, there is a corresponding HPSS function. As the figure shows, HPSS works most efficiently for very large blocksizes - between 1 and 10 MB. Unfortunately, databases typically transfer much smaller amounts of data. In the case of Objectivity/DB, this is a database page, which is limited to a maximum of 64KB. Thus, unless large data volumes were cached on the server side, which brings with it problems with respect to data integrity, such an interface is unlikely to deliver the required performance.

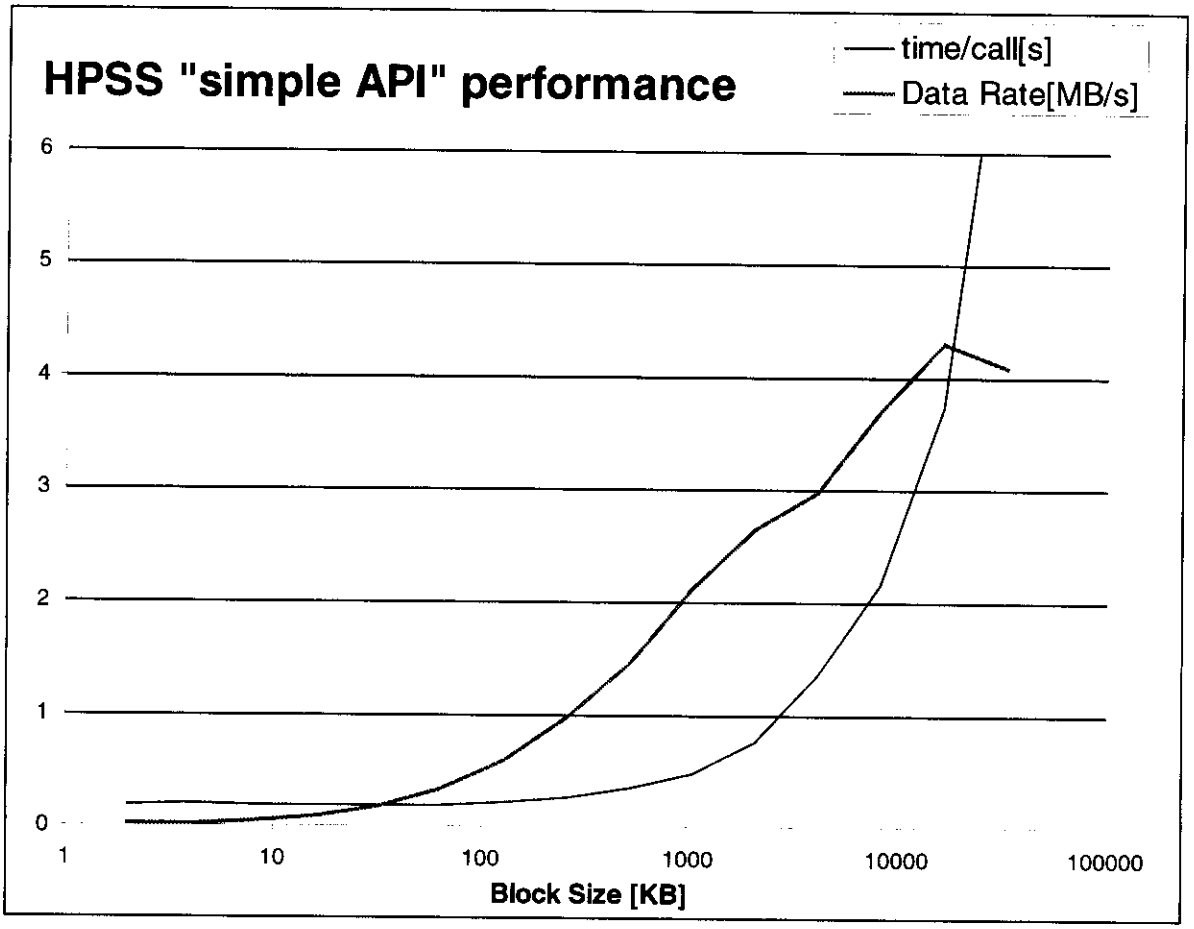


Figure 36 - HPSS Write Performance as a Function of Block Size

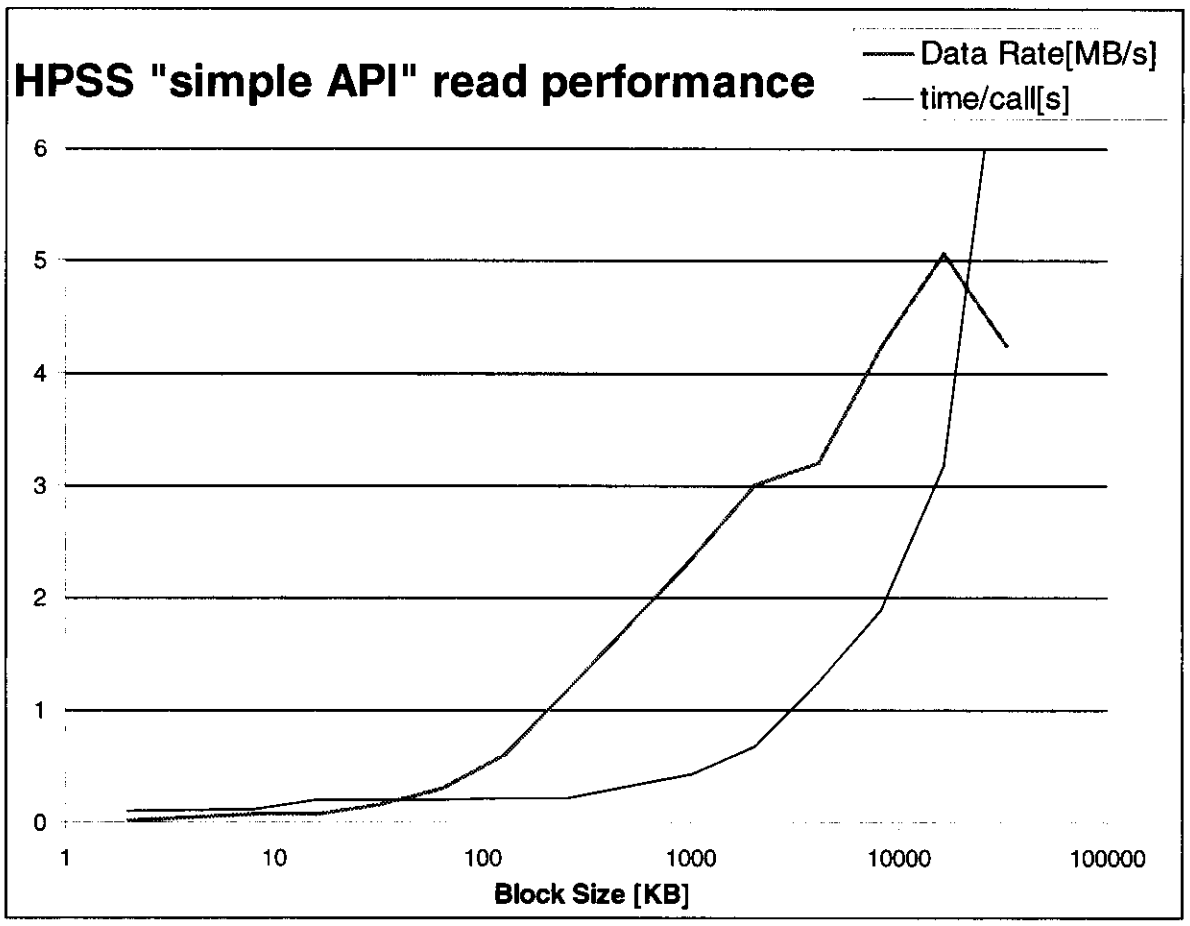


Figure 37 - HPSS Read Performance as a Function of Block Size

In the current prototype, each I/O request to a database residing in HPSS-managed storage involves a significant overhead. Before a data block is transferred, the HPSS client, in this case the Objectivity/DB server, must first contact the HPSS nameserver to obtain the "bitfile ID" of the corresponding file. Having obtained the bitfile ID, it must then communicate with the bitfile server and the data mover to read/write the data. This communication overhead results in a significant performance degradation that suggests that the current interface could not, as anticipated, be used in production.

9.3.2 Alternative Interfaces

We list below the possible interfaces between Objectivity/DB and HPSS.

- Via the HPSS NFS interface,
- Using a future HPSS interface to DFS or DMIG,
- Using the "simple API" (the current interface),
- Using the "advanced API" (permits multiple blocks to be requested),
- Using a DB-faulting mechanism ("staging").

The HPSS-NFS option and the "simple API" both suffer from poor performance and can be ruled out for production systems. The future interface to DFS or DMIG can also be ruled out as a short-term alternative.

The "advanced API" permits clients to transfer multiple blocks without the additional control information being passed to the nameserver and bitfile server. However, in a random-access environment, it is hard to predict which blocks will be read in the future. Furthermore, it is essential that any optimisation does not compromise data integrity. For example, any cached data must be kept consistent across multiple Objectivity/DB server processes or threads. Although the Objectivity/DB client, or rather client application, may have more information about which blocks are likely to be requested in the future, e.g. via the object identifiers in the current event collection, it is unclear whether the use of HPSS for small data transfers is desirable.

An alternative solution would be to use HPSS as a conventional staging system and let Objectivity/DB read/write directly to standard Unix filesystems. This would avoid the performance overheads associated with reading/writing to HPSS-managed disk storage, but would require some space management of the disk pools. However, the existing CERN tape staging software already provides such a capability and is currently being interfaced to HPSS. This can be implemented using the interfaces developed for the Objectivity/DB - HPSS proof of concept prototype and is currently considered the most viable short term solution.

9.3.3 Conclusions on MSS Interface

It is clear that this area needs a significant amount of further study and will be the subject of much attention during the coming year. The activities planned in this area include workshops between IT/ASD and PDP groups and Objectivity experts, visits to SLAC to coordinate activities and compare results and possible implementations, follow-up meetings with Objectivity and the HPSS consortium and so on. The target for a production-quality interface remains the end of 1998 and is scheduled for inclusion in Objectivity/DB V6.0. The use of an interface to a staging system reduces the amount of work required on the Objectivity side, although it is expected that enhancements, such as the ability to pass "hints" from client to server, will be requested in the future.

9.4 CMS Test Beam Experiences

9.4.1 Introduction

A prototype analysis chain was developed in CMS to test Objectivity/DB and other LHC++ components. This software was tested in the H2 test beam for a period of approximately 2 months (August 6th - September 29th). After a few days of running in, the system operated unattended without major problems. A federation of over 60GB was created, with a total of 1250 database files.

This software was also used in the X5 tracker test beam. In this case, a federation of some 25GB in 200 databases was created.

9.4.2 The H2 Test-Beam

The prototype analysis chain tested in the H2 test-beam consisted of the following components:

- Online event data recording:
  - DAQware (ODBMS un-aware),
  - Objectivity/DB formatter (Objy-dependent),
  - Control system (could use ODBMS),
  - CDR (Dependent on Objy Fault Tolerant Option).
- Asynchronous data recording (Objy dependent)
- Offline Data processing
  - Reconstruction framework (ODBMS-based),
  - Interface to simulation,
  - "User" persistent classes (Objy dependent)
- Interactive Analysis Environment
  - Data Browser (Objy dependent)
  - HistOOgrams (ODBMS-based)
  - HistOOgram visualise (ODBMS-aware).

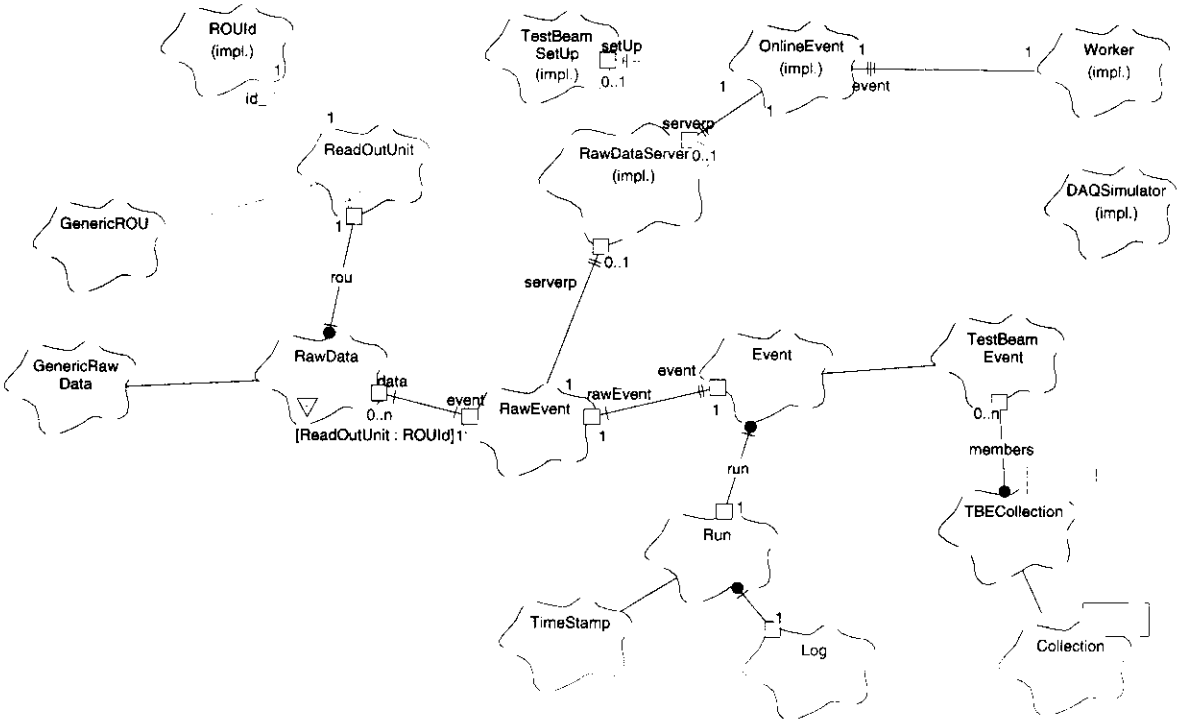


Figure 38 - CMS H2 Test Beam Raw Data Class Diagram

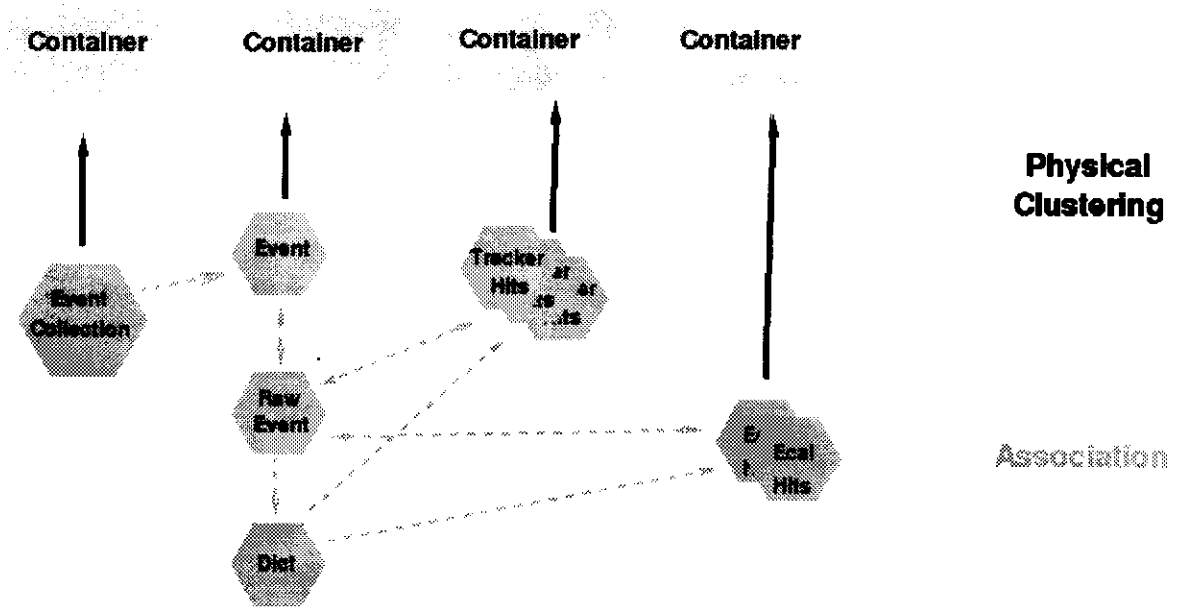


Figure 39 - CMS H2 Test Beam Clustering

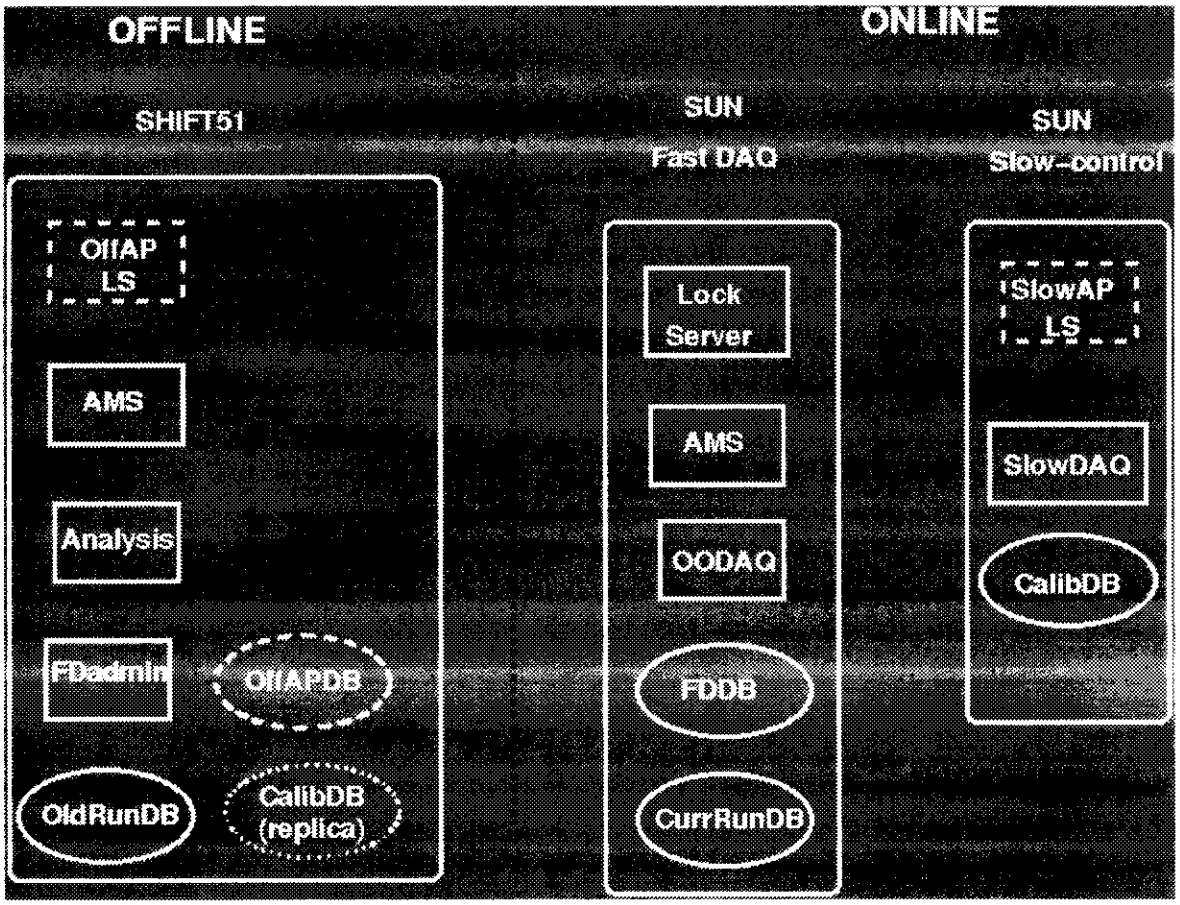


Figure 40 - Test Beam Configuration

After a few days of running in, the system ran essentially unattended without major problems. The only manual operation was to change the output disk every 9GB. Although the system was CPU-bound on object and association creation, a federation of over 60GB (1250 database files) was created and a first analysis performed. The software was reused in the X5 test beam, described below, and is to form part of the common framework developed in 1998.

A number of further developments are planned for 1998, including:

- Versioning,
- Event deep-copy/move,
- Reclustering,
- "User" versioning,
- "User" persistent objects.

A framework for batch analysis of test-beam events was developed, which allowed:

- selection of the input event collection,
- detector reconstruction,
- the storage of selected events in a user collection.

Histogramming is based on the new histOOgram classes from LHC++. User-friendly management of persistent histograms and the usage of generic tags as a potential Ntuple replacement are yet to be tested.

These test-beam activities also allowed testing of development and test federations, based upon the production federation. This was performed using a small script which built a new federation from a reference federation containing the schema, plus a copy of some sample databases from the H2 federation.

9.4.3 The X5B Test-Beam

The CMS X5 OO project is described in detailed in [35]. In common with the H2 OO project, described above, the goal was to create a general framework that could be used in all CMS test-beam areas. This was to include the complete chain from data acquisition to analysis, and hence tested many of the elements of the overall LHC++ strategy.

The X5 Analysis Tool consists of the following components:

- Online (Data Recording)
  - DAQ/Conversion of ZEBRA files,
  - Objectivity/DB reformatter,
  - Central Data Recording,
  - Online Monitoring/Data Quality.
- Offline (Data Processing)
  - Simulation Framework (Interface to GEANT-4)
  - Analysis and Reconstruction Framework
- Interactive Analysis Tools
  - HistOOgrams,
  - HistOOgram Visualiser (HepExplorer/HepInventor)

The Objectivity/DB reformatter performs the following operations:

- Gets the data from the ZEBRA server, using the proxy pattern [36],
- Creates the databases and containers,
- Creates the event structure,
- Populates the databases.

9.4.4 Conclusions on CMS Test Beam Activities

The results from both of these test-beam activities are considered successful by the CMS collaboration. A number of enhancements have been identified for the future, including a scheme for user-friendly management of histograms, and the adoption of the event-tag concept, as an "Ntuple-replacement". Both of these issues are being addressed in the context of LHC++ on the timescale of the 1998 test-beam runs.

9.5 Requirements for 1998 and Beyond

9.5.1 COMPASS

The COMPASS collaboration expects to begin full data taking in the year 2000, with a preliminary run in 1999. The expected aggregate figure for the raw-events sample is 300TB per year.

These data will be processed in parallel with the data acquisition. In this stage the following main steps will be performed for all events: consistency checks of the data sample, a full reconstruction of the tracking systems (and momentum measurement in the two spectrometers), and part of the particle identification. The event information will be combined with the output of calibration runs (as alignment files) and with monitoring data. All the data, both from physics and test triggers, and monitor data from the slow-control system, will be stored in the same federated database. The output of this first processing stage will be some 60TB new information (DST). Typically one full reprocessing stage can be foreseen. The number of physicists involved in this stage will be relatively small - of the order of 10.

Different DST sub-samples will be extracted using only the DST information; these data will be stored on disk, requiring from 3 up to 20TB of disk space, depending on the physics programme. Some 50 concurrent users and many passes through the data are expected.

Due to the aggregate size of the data and the complexity of the analyses, the integration between the database technology and data mining tools is of primary interest for Compass.

9.6 Conclusions

The use of Objectivity/DB in a test-beam activity has been successfully demonstrated, although the data rates involved were clearly much lower than expected at the LHC or for NA45 and COMPASS. Further tests are planned during 1998, including data rates of around 3-4MB/second for NA45.

A proof-of-concept interface between Objectivity/DB and HPSS has been successfully demonstrated and all of the requirements for this version have been met. However, it appears unlikely that the current interface can satisfy the requirements for the production version, particularly in the area of data rates. Current thinking suggests that a simple staging interface is the most appropriate short-term solution to address these performance problems, and such an interface will be developed and tested shortly.

10 Extensions to ODMG-compliant Databases

Although the language bindings defined by the Object Database Management Group offer a significant amount of functionality, it is clear that a general purpose standard cannot - by definition - address the specific needs of a given community. Examples include distributed database administration tools - where database administration is in any case outside the scope of the ODMG - site management tools and application-specific extensions.

In order to facilitate the use of and ODBMS in the HEP environment, a small amount of HEP-specific code has been developed. This code largely falls into two categories:

- Helper classes, distributed in the HepODBMS part of the overall LHC++ framework,
- Specific modules for the IRIS Explorer analysis and visualisation package.

These elements are described in more detail below and in the LHC++ web pages[14].

A good introduction to C++ programming using ODMG-compliant databases can be found in [33]. The standard itself is described in [23].

10.1 HepODBMS Extensions

In order to facilitate the development and support of persistent applications, a small number of helper classes have been developed. These classes, which are distributed as a set of class libraries as part of the overall LHC++ strategy, are referred to as HepODBMS.

The main goals of these classes are to:

- Minimise the dependence on given ODBMS implementation (vendor or release),
- Provide a higher-level interface to ODBMS.
- Minimise the effort involved in porting existing applications that require persistence.

10.2 HepExplorer Modules

HEPExplorer is a set of HEP-specific IRIS Explorer modules, developed in the context of LHC++, which help a physicist set up an environment to analyse experimental data, produce histograms, fit models and prepare data presentation plots using the IRIS Explorer framework. IRIS Explorer itself is a toolkit for visualisation of scientific data, built on top of industry standards such as OpenGL [30] and OpenInventor [31].

HEPExplorer consists of extensions to IRIS Explorer as follows:

[14] See http://wwwinfo.cern.ch/asd/lhc++/index.html.

- HEP-specific modules. Some modules allow for the retrieval, manipulation, fitting and display of histograms stored in an Objectivity/DB database, while other modules allow histograms to be produced out of existing event data.
- Maps[15]. These HEP-specific maps implement simple analysis-related tasks, such as visualise and fit a histogram, produce histograms out of tag data etc.

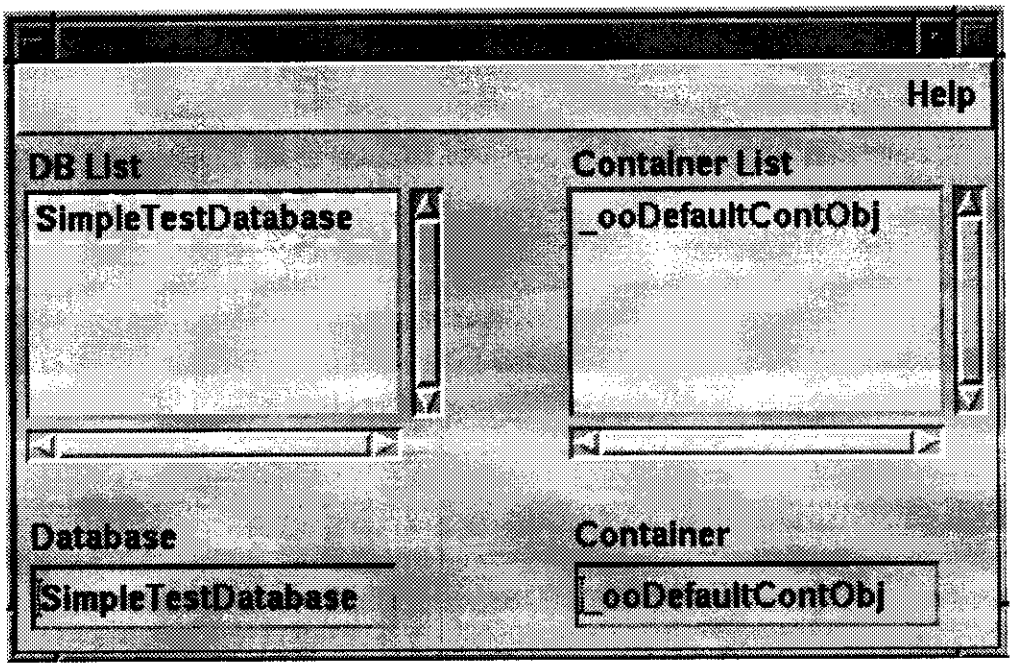


Figure 41 - Prototype of a Database Browser in IRIS Explorer

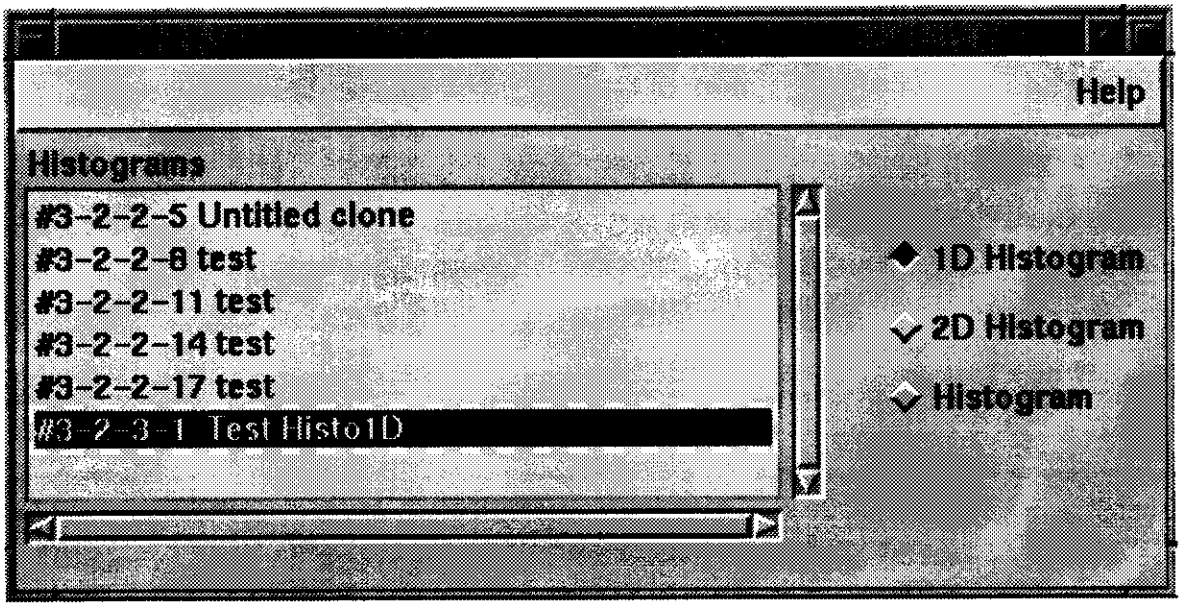


Figure 42 - Prototype Persistent Histogram Browser

[15] In IRIS Explorer terminology, a *map* is an application built out of the basic IRIS Explorer building blocks, or modules. Maps can be built out of modules from a variety of sources, such as standard, public domain and HEP-specific modules.

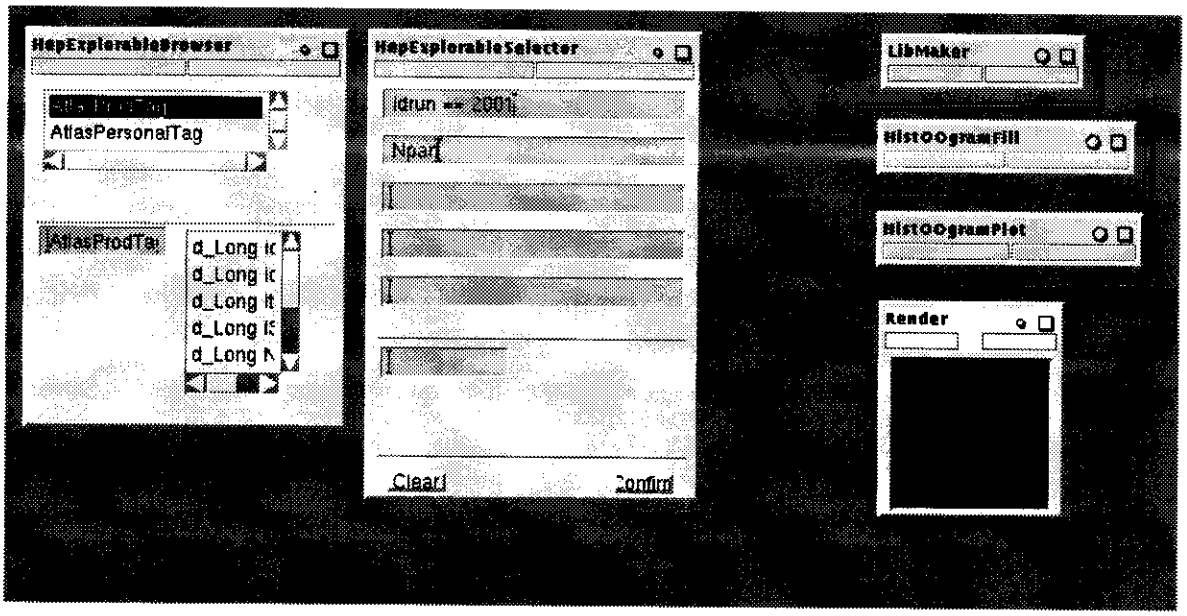


Figure 43 - Example Map Providing "Ntuple/PLOT"-like Functionality

10.3 Calibration Database Prototypes

ODBMS-based calibration databases have been developed both in BaBar and CMS. The basic functionality offered by the two systems is similar, and allows information to be retrieved based upon a "validity time". Calibrations that are stored in the database have a start and end validity time, as shown in the diagram below. The information that is typically stored in such a database includes:

- Electronics calibrations,
- Detector alignments,
- Trigger/Online/Detector configuration,
- Reconstruction adjustable parameters.

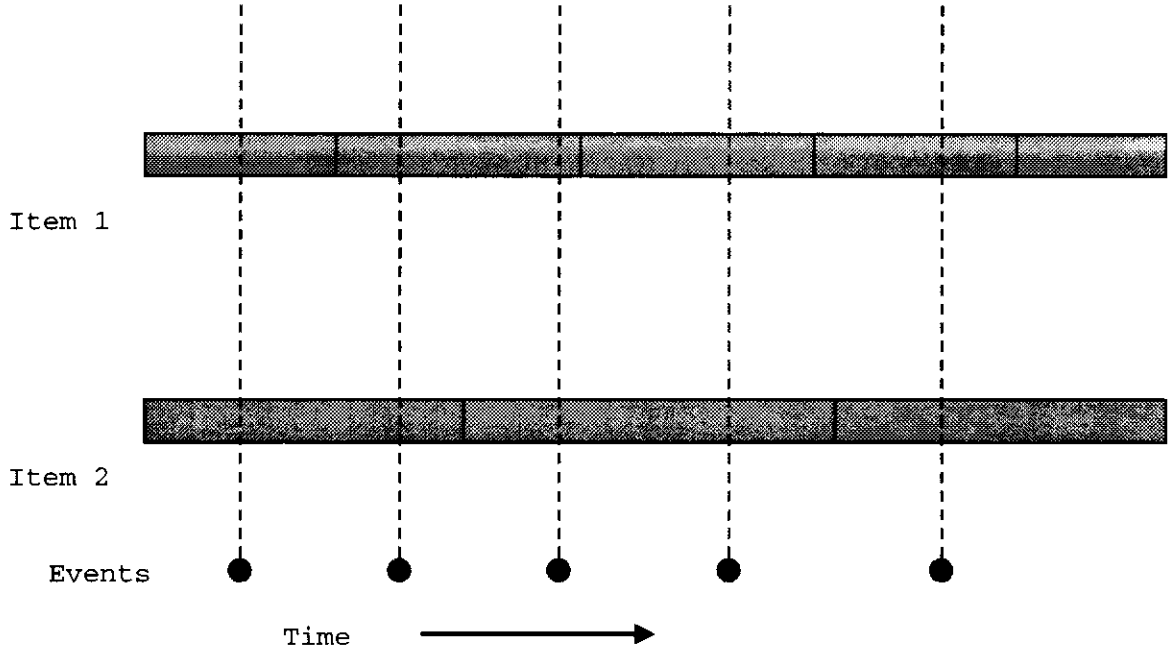


Figure 44 - Calibration Validity Time

It is often the case that improved calibrations are found later, often for a sub-interval of the initial calibration validity range. Thus, one typically retrieves the most recently-inserted constants for the time instant specified.

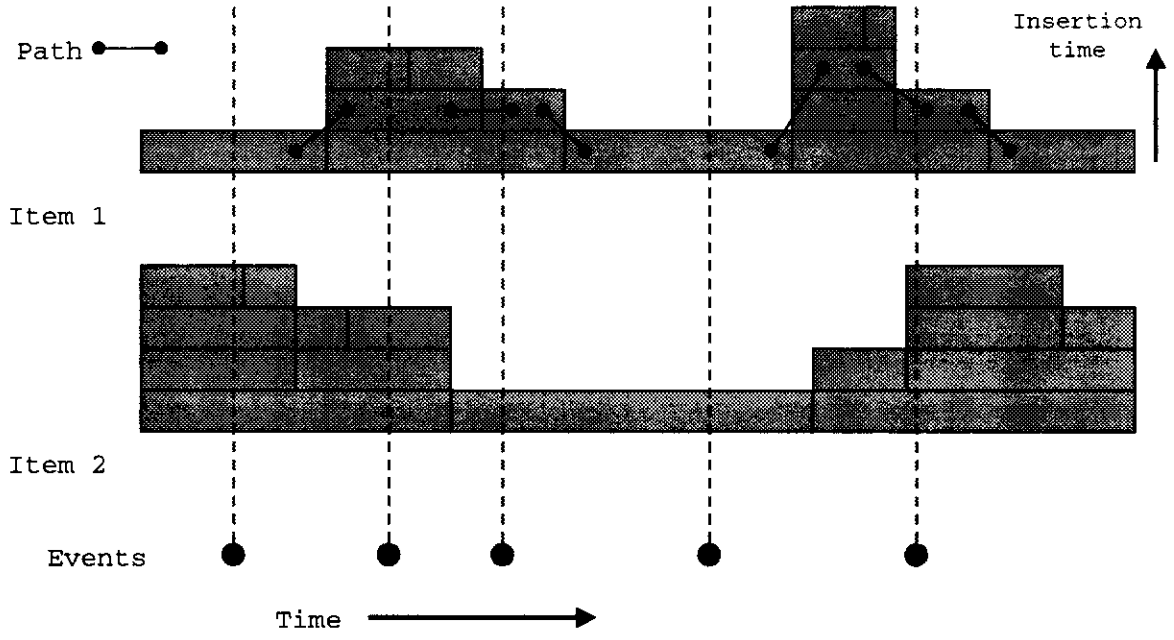


Figure 45 - Multiple Calibrations

10.4 Database Administration Issues

Although the ODMG standard defines a number of language bindings, it does not attempt to define database administration tools or interfaces. Databases such as Objectivity/DB provide both command-line tools and the equivalent programming language interfaces. However, these are typically not well adapted to the fully distributed environment. Hence, a tool for monitoring and administrating an Objectivity/DB federated database has been developed.

A first version of such as tool, named DRO_TOOL, has been built using the Objectivity/DB Java binding. Using this tool, the database administrator is able to observe, control, and manage the basic federated database functionality as well as the autonomous partition and data replication options.

The functionality of this tool is divided in three major groups:
1. configuration,
2. control,
3. statistics.

The configuration group is handles the functionality of the autonomous partition and data replication options. In other words, it allows administrators to create or delete partitions, replicate database images, vary partitions on/offline, resynchronise images and so on.

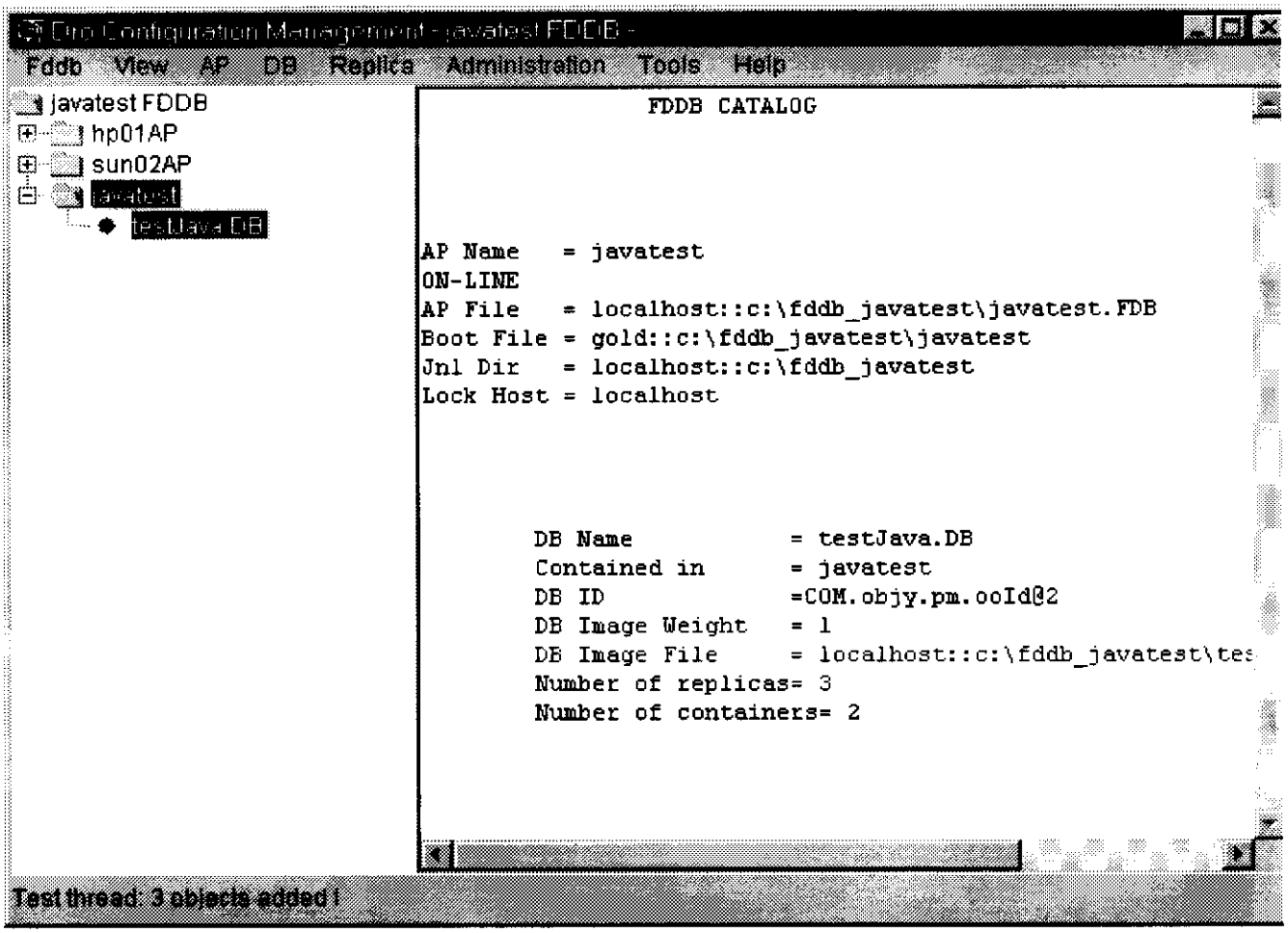


Figure 46 - The Data Replication Management Tool

The control group allows an administrator to monitor and control the database servers. This is performed using the ObjectSpace Voyager product[16], which is also used to permit the tool to run both as a stand-alone application and as an applet in a web browser.

The statistics group offers the possibility to run a number of tests to check data transfer throughput of a given autonomous partition.

[16] To quote ObjectSpace, "Voyager is the first ever 100% Java agent-enhanced Object Request Broker (ORB). It combines the power of mobile autonomous agents and remote method invocation with complete CORBA support and comes complete with distributed services such as directory, persistence, and publish subscribe multicast."

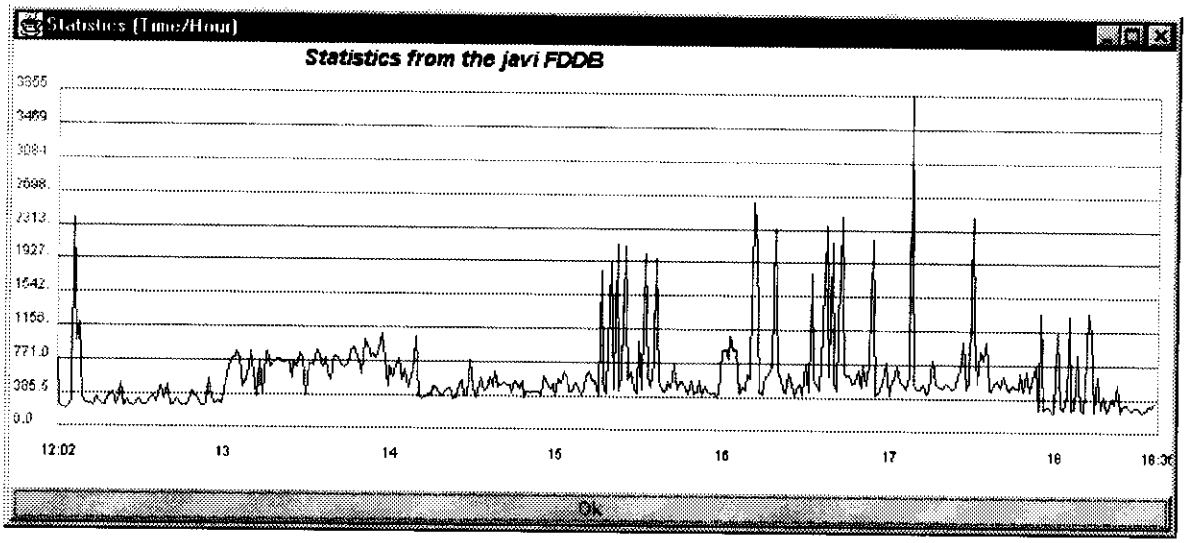


Figure 47 - Performance Statistics from the Management Tool

We note that the current version is very much a prototype, and was initially built as a test of the beta release of the Java binding. Once the Java binding has been officially released, we would expect to design a more powerful tool, based upon the requirements of the experiments and institutes involved. We also plan to evaluate any tools released with future versions of Objectivity/DB, including Java-based data browers, such as the Hudson package, developed by the distributors of Objectivity/DB in Germany[17].

[17] See http://www.micram.de/ot/products/hudson/diver.htm for further details.

11 Tests of the Objectivity/DB Java Binding

11.1 Introduction

Along with many other ODBMS vendors, Objectivity announced a Java binding to their database product during the past year. Although not scheduled for release until February-March 1998, we have made a number of tests of the beta version of the binding, including tests of language heterogeneity, i.e. the possibility to access persistent C++ objects from Java applications and vice-versa.

The Java binding offers us the possibility for tools such as those described above but also opens to us many ways to build flexible distributed architectures. Under the assumption that both C++ and Java analysis applications might exist in the future, it is important to understand any constraints imposed by the database binding and issues such as shared, cross-language schema.

11.2 Impact on application development

As opposed to the C++ binding which relies on the application checking on the status code returned by any database operations, the Java binding uses exception handling. In addition, it supports garbage collection of objects that are no longer reachable.

Persistent Java objects can be clustered in two types of containers:

1. Garbage-Collectable containers: used to store directed graphs of objects that represent composite objects. Applications do not need to delete objects that cease to be part of the graph.
2. Non garbage-collectable containers: used to store objects that are not necessarily connected to others.

In a mixed-language environment, it is clearly necessary to use features, such as non-garbage collected containers, which are supported by all languages concerned.

The Java binding also provides a number of clustering classes. Two types of clustering are supported:

1. Explicit clustering: calling the *cluster* method. As in C++ binding, an object can be clustered close to another object, in a container or in a database.
2. Implicit clustering: every transaction has a clustering strategy that performs implicit clustering for objects that are made persistent within that transaction. A user-defined clustering strategy can be provided, such as where an object is assigned to a random container in a pool of containers maintained by the database that contains the object.

As opposed to the C++ binding, where the schema of persistent capable classes must first be defined, the Java binding permits the definition and creation of persistent classes at runtime. The type number allocation is therefore necessarily dynamic, solving the schema handling problems described in section 8.8.

The following figure shows the comparison in application development between the two bindings:

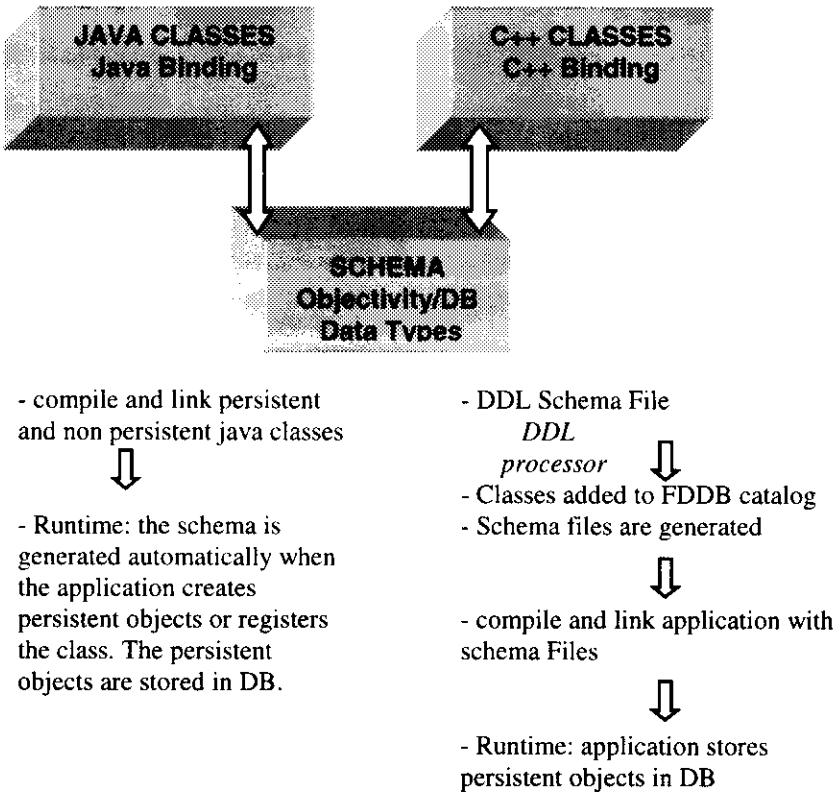


Figure 48 - Schema Definition in C++ and Java

11.3 Impact on DB/types

As mentioned in the Object Database Standard ODMG 2.0, the Java Binding does not introduce new constructs specific to the database: the binding is perceived as part of the Java language according to the following principle.

"The ODMG Java Binding is based on one fundamental principle: the programmer should perceive the binding as a single language for expressing both database and programming operations, not two separate languages with arbitrary boundaries between them. " [23]

Even though the binding fully accomplishes this requirement from the standard, mixed-language applications still need to take into account the fact that not all types in Java have a one to one mapping to those in C++.

In the beta version of the binding, the *oojVarray* is not correctly mapped to its C++ equivalent, *ooVarray*. This should be corrected in the production release. In addition, there is currently no mapping between e.g. STL-based collection classes in C++ and Java collections. We understand that this issue will be addressed in a future release of Objectivity/DB.

11.4 Tests of Java Agents

The use of Java offers a number of interesting opportunities that extend the traditional client-server architecture of Objectivity/DB. Not only can multi-tier applications be readily implemented, but the use of Java agents provides a simple mechanism whereby the query can be moved to the data, execute and then move the results back to the host from which the user issued the query. For example, one could communicate from an applet activated from a Web browser to a server that in turn communicates with the database. The applet itself would not need to be linked against Objectivity/DB, nor would this software need to be installed on the client computer.

11.5 Summary

The recently announced Java binding to Objectivity/DB appears to be well suited to the development of tools for database management and configuration. The potential offered by mobile Java agents and Java's in-built network support is clearly worthy of detailed investigation. Our main requirement with respect to the Java binding is that of full inter-language operability with C++, which in turn requires a convenient mapping of the data types of the two languages.

12 Objectivity/DB Enhancement Requests

Our experience with Objectivity/DB has, as predicted, resulted in a number of enhancement requests. These requests are fed-back to Objectivity by means of the regular RD45 workshops and the Objectivity user meetings. The list of outstanding enhancements is regularly reviewed, enabling us to follow up on these issues. A number of key enhancements have already been addressed, such as the need for STL-based persistent collection classes. Others are being worked on, such as an interface between Objectivity/DB and HPSS. It is our understanding that our main enhancement requests will all be addressed on an appropriate timescale; most, if not all, should be delivered in time for BaBar/COMPASS production in 1999. Clearly, we will continue to come up with new requirements, which we will prioritise and feedback to Objectivity. The main enhancement requests are discussed in more detail below.

12.1 Support for STL-based Collection Classes

Previous versions of Objectivity/DB supported - in common with a number of other ODBMS products - a persistent version of the Rogue Wave *Tools.h++* collection classes. However, with the emergence of the STL, the need for STL-compliant persistent collections became clear. Such collections have been added to the C++ binding of version 2.0 of the ODMG specification. We therefore requested that Objectivity support such collection classes and suggested that no further releases of their persistent version of the Rogue Wave classes were required. As of the 5.0 release of Objectivity/DB, STL-based collection classes, using the ObjectSpace implementation, are supported as part of the product. The previous Rogue Wave classes have been dropped.

12.2 ODBMS to MSS Coupling

Although, as described in [12], there is reason to be optimistic concerning the evolution of storage capacity versus cost, it is still unlikely that multi-PB disk farms will be either affordable or practical at the time of LHC startup. To solve this problem, the RD45 collaboration has studied a number of possible ways whereby an ODBMS, and Objectivity/DB in particular, could be interfaced to a Mass Storage System. The MSS of choice at CERN for the foreseeable future is the High Performance Storage Server, HPSS. A training course on HPSS was held at CERN during October 1996, to which an Objectivity engineer participated. As a result of this course, a proposal for integrating the Objectivity/DB server (AMS) to HPSS via the HPSS client API was made. This proposal was further discussed between members of the HEP community, representatives of the HPSS consortium and Objectivity at a meeting at Objectivity's headquarters in May 1997. As a result of this meeting, Objectivity committed to producing a proof-of-concept prototype by the time of SuperComputing 97, held in San Jose in November 1997. The requirements for the prototype were as follows:

- The Objectivity - HPSS proof-of-concept prototype should provide transparent client access to a federation stored in HPSS-managed storage,
- The prototype should demonstrate client-transparent migration/staging of databases to/from tertiary storage,
- A demonstration of sequential and random read & write access and creation/deletion of databases should be made,
- A demonstration of access from Unix and NT clients should be performed.

Production requirements for the Objectivity/DB - HPSS coupling (see section 6.2.2 on page 26) come from a number of experiments, including BaBar at SLAC, NA45 and COMPASS at CERN, and of course the LHC experiments themselves. However, it is the timescales of the pre-LHC experiments that dictate when a production version must be ready. We have requested that a product be shipped no later than Q4 1998 - it being understood that extensive testing would be performed at a number of HEP sites during the latter half of 1998.

12.3 Architectural Changes to Support VLDBs

The current architecture of Objectivity/DB comprises:

- A *federation*, which consists of up to 2^{16} - 1 distributed databases, sharing consistent schema and permitting transparent cross-database references,
- *Databases*, which today map to files, which are in turn made up of 2^{15} - 1 containers,
- *Containers*, composed of up to 2^{16} - 1 logical pages,
- *Pages*, on which the objects themselves are stored (up to 2^{16} slots).

The database page size is a constant for the entire federation, and is limited to 2^{16} bytes.

Theoretically, this architecture permits federations of up to 10^{19} bytes. However, a number of practical limitations mean that such sizes will never be achieved. The most important limitation is that of the filesize. Here we feel that 100GB per database (file) is probably an upper limit - today 1-10GB is perhaps more reasonable. As a rule-of-thumb, we feel that it should be possible to migrate/recall a complete file (database) in 10^2 - 10^3 seconds. A file of 100GB would require an overall, if parallelised, I/O bandwidth of 1GB/second to reload in 100 seconds, whereas a 10GB file would require only 10MB/second to reload in 1000 seconds.

Using a maximum file/database size of 100GB - derived from the practical limits given above - federations of 6.5PB are then possible. This would not, however, be sufficient to store all data from a single LHC experiment. We have therefore asked for architectural changes that permit 100PB federations, without imposing arbitrary constraints, such as requiring containers or databases to be full to reach this limit.

12.4 Schema Handling Enhancements

Enhancements to the way in which the schema for persistent C++ classes are handled are required such that it is easy and transparent to develop applications across multiple federations. In other words, the developer should be able to build an application using a test federation, and not the production federation of a given experiment. This would require, for example, that no type numbers are hard-coded into the header files produced by the DDL processor. An acceptable solution would be to adopt a similar mechanism to that employed in the Java binding, where the type number of determined at run-time. Such changes should be compatible with currently supported features, such as support for named schema, classes of the same name, but in different named schema and for schema evolution.

12.5 Access Control Support

In the current version of Objectivity/DB, access control, based on client credentials, is not supported. It is a requirement that such support be added to a future version of Objectivity/DB. Such access control must work consistently across the entire federation, be supported by both language bindings and tools and support both rôle-based (e.g. DBA) and user-based activities. Given the difficulty of implementing a consistent authentification scheme on all relevant nodes in a federation, exits, e.g. at database open time, where site-specific code may be called would be a valid, if not preferred, solution.

12.6 ODMG Compliance

The C++ binding of Objectivity/DB is not fully ODMG-compliant in a number of areas. For example, the ODMG specifies methods *d_activate()* and *d_deactivate()*, which are called when an object enters or leaves scope. It is a requirement that fully ODMG-compliant bindings be provided for all of the languages of interest to HEP (C++, Java), although vendor extensions, for the purpose of performance, are acceptable if clearly marked as such. The Objectivity/DB documentation and training material should be based on the corresponding ODMG language binding.

12.7 Support for the Linux Operating System

Interest in running a version of Unix on cheap commodity processors - i.e. Intel Pentium and similar - has grown considerably over the past two years. The Linux operating system has clearly emerged as the preferred Unix for PCs within the HEP community. This has resulted in a number of informal requests to Objectivity to include Linux in the list of supported platforms. At the time of writing, it is our understanding that Objectivity will provide support for Linux in a future release for a well-defined operating system and compiler combination, such as Red Hat 4.1 and g++ 2.7.2.

13 Standards Activities

In the context of RD45, CERN has *associate membership* of the Object Management Group (OMG) and is a *reviewer member* of the Object Database Management Group (ODMG). CERN is also represented in the IEEE Computer Society Executive Committee on Mass Storage, which is the body to which the various standards sub-groups report. As in previous years, CERN has only participated actively in the ODMG.

13.1 ODMG-related Activities

During the past year, the ODMG released V2.0 of its book, defining the object model for object databases and the various language bindings. This version of the standard is viewed as being a significant improvement over previous versions. The main changes are:

- The adoption of STL collection classes,
- A new Java binding.

The release version 2.0 marked a turning point in the ODMG. The working groups, which previously met 8 times per year, now meet less frequently: as little as three times a year for the C++ working group. Effort will continue on issues such as Java, but many of the other bindings can now be considered more or less stable. Other changes are being discussed, such as a broadening of the ODMG charter to include persistent objects for relational mappings and application servers, rather than just object databases.

Although most of the ODMG meetings are held in the US, a meeting was held in July 1997 in Annecy. It was at this meeting that the priorities for post-V2.0 work were discussed. Of the enhancements proposed by the voting members[18], the main priorities, from the CERN point of view, were:

- Support for *int64* type,
- Collection classes for the Java binding, based on the ObjectSpace generic collection library for Java (JGL) collection classes,
- Support for distributed databases in the ODMG model,
- A proposal for changes to the Java Virtual Machine (JVM) to be submitted to JavaSoft,
- Support for typed collections (templates) for the Java binding.

Given the reduction in ODMG activities, it is felt that CERN should reduce its participation accordingly, attending approximately 1 meeting per year, rather than 2 out of 4, as has been the case so far. However, it should be noted that the benefits of involvement in the ODMG go beyond the possibility of influencing the standard itself. Firstly, they allow us to have access to information concerning new ODMG features before they appear

[18] Under ODMG rules, voting members are those that ship a commercial ODBMS, or have implemented or announced implementation of one of the ODMG bindings. Although this latter clause - a recent revision - would theoretically permit CERN to acquire voting rights, we would then be required to devote significantly more resources to ODMG activities than can currently be envisaged.

in the published standard, which allows us to put pressure on the suppliers to implement the new features in a timely manner. In addition, the ODMG meetings offer an excellent opportunity to meet developers from the various database vendors and also allow the work at CERN to be more widely exposed. For example, the joint ODMG/JavaSoft press release concerning the ODMG binding for Java contained statements from many database vendors, but only a single end-user site, namely CERN.

14 General Database Activities

In addition to the work-items related to the LCB milestones and recommendations, the following activities are worthy of note.

14.1 Objectivity/DB Workshops

As in previous years, we have held a series of RD45 workshops at CERN. These have been well attended by members of experiments both at CERN and outside, and also by people from other (non-HEP) ODBMS-based projects. These workshops have been extremely useful for discussing and sharing ideas and experiences between different groups, and for feeding back information on enhancement requests to Objectivity. However, as the number of participants has grown, the workshops have evolved from informal working sessions to more formal presentations. We have therefore started a series of mini-workshops, focussed on very specific issues. The first such workshop, discussing event collections and related issues, took place at CERN from February 19-25 1998.

14.2 Objectivity/DB User Meeting

As in previous years, an Objectivity/DB Developers' Conference was held in Santa Clara in May 1997. This conference, which offers an excellent opportunity to meet other developers, e.g. working on the MOTOROLA Iridium[19] project or the Sloan Digital Sky Survey[20], and Objectivity engineers. As in the past, a paper on RD45 was presented, giving both a brief status of the project and a list of the main outstanding enhancement requests. At the conference, Objectivity announced their plans to support an interface to the HPSS system, which had only formally been requested a few days previously, at a joint meeting between various HEP laboratories and representatives from the HPSS consortium and Objectivity.

14.3 Licensing Issues

Based on larger volumes, we have now been able to obtain even better discounts than in the past. Similar discounts are also available for other HEP laboratories, several of which have acquired licenses for their own research programme (BNL, DESY, KEK, SLAC etc.) It has recently been agreed that the funding of license acquisition and associated maintenance costs for CERN experiments and collaborating institutes will be handled by the annual COCOTIME allocation of computing infrastructure. Experiments will be asked to estimate their requirements for the coming year and the necessary funds provided centrally through CERN. An amendment to the CERN contract with Objectivity has been negotiated, such that licenses may float across all institutes collaborating in the CERN research programme, within the limit of the total number of licenses available. The possibility of unlimited usage on the CERN site is being investigated.

[19] See the Motorola Web site or http://www.mot.com/General/Events/InteractiveTelecom/satcom/Iridium.html.
[20] See http://www-sdss.fnal.gov:8000/.

14.4 Objectivity/DB Support

During the past year, Objectivity introduced Web-based access to their support team. This allows registered users - a few people per experiment and members of the IT/ASD group - to query the internal problem database and see much more detailed information on the status of problem reports than was previously possible. This web site complements the standard e-mail support and the on-site consultancy organised principally via the RD45 workshops.

15 Other Database Developments

The Object Database market continued to evolve over the past 12 months. Of the events that have occurred, we believe the following to be of most significance:

- The size of the market is estimated to have grown to $150-$200M. The growth rate is estimated to be 45% - exceeding the growth rate of the RDBMS vendors. Should this rate remain constant, then the total market will exceed $1B in around 2001/2002.
- The emergence of the ODMG Java binding for Object Databases, and compliant bindings from a number of vendors.
- O_2 was taken over by the US-based Unidata. Unidata itself was subsequently taken over by VMARK[21], a competitor of Unidata. Both Unidata and VMARK have traditionally competed in the RDBMS market. It appears that they will follow the strategy of offering a "universal server", in common with other RDBMS vendors. The new company is now called Ardent Software[22].
- POET has forged strategic links with both Microsoft and Novell. Their product now has a number of features that only work on Microsoft platforms (NT, 95) and rumours continue to circulate that they will be acquired by Microsoft.
- Computer Associates delivered its "Jasmine" ODBMS product, based upon work by Fujitsu/ICL. This is considered important in that it is the first time that a large software company implicitly acknowledges the importance of the ODBMS market.
- Objectivity refocussed on its "core business".

During the second half of 1996 and 1997, Objectivity was apparently gearing up to go public. This manifested itself by a rapid increase in personnel and a new management structure. At the same time, a new division, AZIZA, was formed. The AZIZA division focussed on an Objectivity/DB-based web management tool of the same name. Although this tool had many interesting features in its own right and had the additional benefit of stress-testing many of the newer features in Objectivity/DB, it appeared to be diverting too many resources from the base product. During the second half of the year, the company split off the AZIZA division and restructured to focus on its database product. We believe that this restructuring was both important for CERN/HEP and necessary for the company. We have seen a marked improvement in response to our requests for enhancements since the change - the company policy is now clearly to focus on the high-end and become the Object Database of choice for this market segment. Clearly, the company will have a challenging year or two ahead, to truly establish itself as the leader in this market. Here, Objectivity as a company understand that satisfying the requirements of the HEP community - such as for an interface to HPSS - give them a significant advantage over their competitors.

[21] See, for example, the letter to customers available via http://www.vmark.com/merger/customer.html.
[22] See http://www.ardentsoftware.com.

16 Use Of Objectivity/DB in HEP

We list below the experiments that are currently using or testing Objectivity/DB-based solutions.

16.1 AMS

The Alpha Magnetic Spectrometer[23] is an experiment that will take data first on the NASA space shuttle - launch date May 1998 - and later on the International Space Station. The physics goals of the experiment are to perform an antimatter and dark matter search. The AMS collaboration has been using Objectivity/DB in test and plan to use it to store their production data, slow control parameters and NASA auxiliary data.

16.2 ALEPH

The Aleph collaboration has recently started an exercise whereby their mini-DST will be copied from its existing ADAMO-based format to Objectivity/DB. The purpose of the exercise is to gain experience with more "modern" analysis tools, i.e. those currently proposed as part of the LHC++ strategy.

16.3 ALICE

The ALICE offline team is currently focussing on GEANT-4. At the time being, it has no explicit activities in the context of RD45. During 1998, it will begin to study Objectivity/DB-based solution, but again only in the context of GEANT-4.

16.4 ATLAS

The ATLAS collaboration is developing a number of prototype applications using Objectivity/DB in both on- and off-line communities. These include providing access to GEANT-3 simulation data stored in Objectivity/DB and will naturally evolve to storing the hits, digits, etc. from ATLAS GEANT-4 simulation and the results of the reconstruction in Objectivity/DB. In addition, attempts are being made to store the detector description in Objectivity/DB, and to develop various online applications, such as a run booking system and calibration database on top of the system.

16.5 BaBar

The BaBar experiment at SLAC, due to start taking data in 1999, plan to use a combination of Objectivity/DB and HPSS in which to store their data. They currently expect to record some 200TB of data per year, all of which will be stored as persistent objects in an

[23] See http://hpl3tri1.cern.ch/ for more details.

Objectivity/DB federated database. The majority of the associated storage would be managed by HPSS.

16.6 BELLE

The Belle experiment at KEK starts taking data in fall of 1998 and plans to use Objectivity/DB to store the detector constants. They also hope to store mini/micro DST using Objectivity for rapid data analysis. Future mass storage plans are currently being developed in conjunction with the KEK computing centre.

16.7 CDF

The CDF Collaboration is currently evaluating Objectivity/DB as a possible data management system for their RUN-II, which is expected to begin in 2000 and collect about 450 TB of data within a two year period. Prototype databases containing the RUN-I data have been created, and valuable experience has been gained in the area of optimisation of the database parameters depending on the data model used. The data model proposed for RUN-II is similar to that of ATLAS or Babar, with all data (or, initially, just the reconstructed information) stored as persistent objects in an Objectivity/DB federated database. It is anticipated that an HSM (HPSS is proposed at present) will be used to manage the tape storage. An interface between Objectivity/DB objects and the currently used YBOS (TRYBOS) records has been developed and tested, as many reconstruction algorithms will still be FORTRAN based. At present, a prototype database that realises this new data model is being developed. The CDF Collaboration is expected to make the final decision in May 1998.

16.8 CHORUS

The CHORUS collaboration is using Objectivity/DB for an online emulsion scanning database. There are plans to use the same application at a number of collaborating institutes. As a by-product of this activity, they will also evaluate Objectivity/DB as a potential solution for the proposed TOSCA experiment.

16.9 CMS

The CMS collaboration is using Objectivity/DB for a number of prototype applications, including the test beam activities, discussed earlier in this report. As with ATLAS, the current baseline assumption is that the event and associated data will be stored as persistent objects in an object database, combined with a mass storage system. Currently, it is assumed that this will be Objectivity/DB together with HPSS.

16.10 COMPASS

The COMPASS collaboration plans to use Objectivity/DB to store all of its experimental data (300TB raw data per year). In the framework of the prototyping of the COMPASS

computing farm, a test at full data rate (35 MB/s) will be performed at the end of 1998. Data from the central data recording will be sent to a federated database and the integration with HPSS will be also tested.

16.11 LHCb

The LHCb collaboration are currently writing their technical proposal and notes. During the current year, the main emphasis will be on GEANT-4 related issues, including the design of the geometry database, the implementation of GEANT-4 within LHCb and designing high-level event classes. General topics for 1998 include event collections, replication and remote access to data and C++ and Objectivity/DB training.

16.12 LEP Data Archive

A project has recently been proposed[24] whereby existing LEP data would be archived for a period of some 30 years. Over such a long period of time, it is assumed that little of today's computing environment would remain. In particular, it has been assumed that the current CERN Program Library, existing operating systems and the Fortran programming language will not longer exist. It is estimated that a few TB of data per experiment will need to be converted and stored, giving a total data volume of perhaps 20TB. It is currently assumed that the data will be stored in Objectivity/DB and that a demonstration of an analysis against the database made.

16.13 NA45

NA45 have been using Objectivity/DB in production since early 1996. A number of production runs have been performed, with a total data volume of some 30GB. For 1998, their plans are to make tests of Objectivity/DB together with central data recording and HPSS, in preparation for their 1999 data taking run, where some 30-50TB of data are expected.

16.14 NA48

NA48 have a detector configuration management application based on Objectivity/DB. Recently, a new project has been initiated to optimise access to physics data by storing compact micro-DST information and perhaps more in Objectivity/DB. This project is similar in conception to that undertaken by ZEUS.

16.15 RHIC Experiments

The RHIC experiments at Brookhaven plan to adopt a common strategy for their data storage. The current plan is that this will be based, as at other laboratories, on a combination of Objectivity/DB and HPSS. Experiments involved include BRAHMS,

[24] See http://wwwinfo.cern.ch/s/sticklan/www/archive/ for more details.

PHENIX, PHOBOS and STAR. Data volumes for both PHENIX and STAR are expected to be around 200-300TB/year.

16.16 ZEUS

The ZEUS experiment has built a tag database on Objectivity/DB, which has been used in production since the end of 1997. This database has been built from the physics data in the ADAMO [24] database. The new system offers considerably more flexibility than was possible in the past. For example, instead of selecting on a combination of 128 bits, the definition of which changed with time, the user is now able to select using a more meaningful predicate string. In addition, predefined "named" samples can be defined, convenient for frequently used samples. The new system has proved popular with the physicists, not least as it offers improved performance - a direct result of reading only the required data.

17 Future Activities

17.1 Executive Summary

In the original RD45 project proposal (CERN/DRDC 94-50), three phases were anticipated. The first phase, which lasted approximately 6 months and at the end of which time an interim status report was made to the LCRB, was devoted to obtaining a better understanding of the problem and a preliminary list of requirements. These requirements are listed in the status report for the first year [11]. This was followed by a second stage, in which detailed prototyping and performance comparisons were undertaken [4] [5] [6] [7]. It is the opinion of the RD45 collaboration that both of these two phases have now been completed. Furthermore, we believe that the project has achieved its stated goals of identifying a solution to the object persistence problems of the LHC experiments. We suggest that the project now enter the third phase foreseen at the time of the project proposal, namely that of implementation. This phase is expected to consist of two elements:

1. Production services, based upon the possible solutions evaluated during phase 2 of the project, i.e. Objectivity/DB, HPSS and the HepODBMS class libraries,
2. Further R&D into areas where questions still remain. The list of such issues is currently being compiled together with the LHC experiments.

17.2 Introduction

The LCB/RD45 project has been studying issues related to the problems of object persistency since 1995. As an LCB project, the focus has clearly been on the needs of the LHC experiments. As described in the various RD45 status reports, a strategy has been built up, in close collaboration with the experiments, based on the use of two commercial components - namely Objectivity/DB and HPSS - with a small number of HEP-specific extensions. This strategy has been adopted by a number of pre-LHC experiments, including COMPASS and NA45 at CERN, BaBar at SLAC and the RHIC experiments at Brookhaven. In addition, there are numerous projects, both at CERN and outside, based upon Objectivity/DB alone. These include activities in CHORUS, NA45, NA48 and the LEP experiments. Finally, as agreed at the 1997 COCOTIME review, there is a need for production Objectivity/DB federations in 1998 for both ATLAS and CMS: systems on which to run these services are being acquired now by IT/PDP.

In summary, there are many requests for general-purpose production services at CERN, based upon Objectivity/DB. We discuss below how these services could be established and how the research issues related to the object persistency services for the LHC experiments could be addressed.

17.3 Production Services

We propose that IT division begin to offer data management services, based upon Objectivity/DB and HPSS. IT/ASD group offer ODBMS services, similar to the current services based on ORACLE for RDBMS applications. IT/PDP would be responsible for issues related to HPSS and the data servers on which the Objectivity/DB server and HPSS client would run. Clearly, a detailed service definition needs to be drawn up by IT/ASD and PDP groups. For the purposes of this document, we focus on those issues that will be handled by IT/ASD group.

IT/ASD group would:

- Acquire and install the Objectivity/DB software on the central services at CERN, e.g. AFS and NICE,
- Handle license management, software distribution, documentation and related issues,
- Provide user-support and consultancy services to users and experiments building and/or deploying applications based upon Objectivity/DB,
- Provide both user guides and reference manuals for any applications or tools that are developed, such as the existing DBA tools, the HepODBMS class library and the prototype calibration database,
- Implement a problem tracking system, in common with the rest of LHC++,
- Organise e-mail distribution lists of the distribution of general information and for detailed discussions (e.g. the event collection discussion list),
- Continue to organise regular, video-conferenced meetings and workshops with the user community to understand their requirements,
- Report on an annual basis to the LCB and other bodies (FOCUS, HEP-CCC, CHEP etc.) on the status of these services,
- Ensure consistency between Objectivity/DB-based services and those based upon other components of LHC++,
- Provide and support HEP-specific extensions as required, such as the HepODBMS class library, database browsers for use with IRIS Explorer and stand-alone, database configuration and management tools etc.,
- Liase with Objectivity concerning release schedule, bug fixes and enhancement requests,
- Participate in Objectivity/DB User Group meetings.

17.4 Research Activities

We believe that the RD45 project has attained its primary goal of identifying a solution to the problem object persistency. Clearly, there are many issues that still need to be pursued, such as the outstanding list of enhancement requests, including the production version of the Objectivity/DB - HPSS interface. In addition, there are many associated issues that have not yet been addressed, such as methods of optimising data access and management in

a fully-distributed environment, the impact on networking costs, new technologies such as agents, and so forth. Furthermore, neither Objectivity/DB nor HPSS have yet proven themselves in production environments with data volumes of the order of many hundreds of TB. In this respect, we anticipate that much will be learnt from the experiences of high-data volume, pre-LHC experiments such as BaBar, COMPASS, CDF, D0, PHENIX, STAR and others.

Many of these issues are best addressed as part of the general Objectivity/DB services, such as those proposed. Others require further research and development, but perhaps on somewhat longer timescales than in the past.

17.5 Summary

We believe that the RD45 project will have attained its primary goal of identifying a solution to the problem of providing object persistence services for event and other data for the LHC experiments by the time of the April 1998 LCB review. We recommend the setting up of production, ODBMS-based services, using the technologies and solutions identified in this work. Further investigation into topics identified by the LHC experiments and/or the LCB would continue to be addressed on a timescale compatible with the needs of the experiments and of the available resources.

18 Proposed Milestones for 1998-1999

As described above, we believe that the main research phase of the RD45 project, namely to research into and propose solutions to the "object persistency" problems of the LHC experiments, to have been completed. A logical next phase would be the setting up and running of production services for general CERN use. Even though this phase would be labelled as "production", there would clearly be on-going developments, continued discussion with Objectivity on enhancement requests, further prototyping of ODBMS-based applications and class libraries and so forth.

The key achievements of the RD45 project over the past years have been as follows:

1995: evaluation of ODMG ODL, ODBMS+MSS prognosis, proposal of ODBMS+MSS as "solution"
1996: impact of using an ODBMS, evaluation of ODBMS features, performance comparison, risk analysis
1997: demonstration of ODBMS in data-taking, simulation, reconstruction and analysis scenarios up to 1TB

We see the natural evolution of this trend as being:

1998: together with IT/PDP group, set up production services up to 10TB
1999: production usage by BaBar, BELLE, COMPASS, NA45, BRAHMS, PHENIX, PHOBOS, STAR, ZEUS ...

Over the past years, the fraction of RD45's activities devoted to production-related work has increased significantly. There are now several experiments at CERN using Objectivity/DB in production, and both NA45 and COMPASS intend to use it, together with HPSS, for their production runs in 1999. Preparing for these runs will be an important component of the activities of the coming year. However, we believe that these are best covered outside of an R&D project, e.g. as part of a standard service offered by IT division, in collaboration with the experiments.

We therefore suggest the following activities for the next 18 months:

1. Provide, together with IT/PDP group, production data management services based on Objectivity/DB and HPSS with sufficient capacity to solve the requirements of ATLAS and CMS test-beam and simulation needs, and COMPASS and NA45 tests for their '99 data-taking runs.
2. Develop and provide appropriate database administration tools, (meta-)data browsers and data import/export facilities, as required for the above.
3. Develop and provide production versions of the HepODBMS class libraries, including reference and end-user guides.

4. Continue research, based on input and use cases from the LHC collaborations, in the following areas to provide results in time for the next versions of the ATLAS and CMS Computing Technical Proposals (2nd half of 1999):
   - Database usage over a wide area network,
   - Clustering/reclustering strategies,
   - Interface to MSS.

19 Conclusions

The RD45 project was approved in 1995 to investigate and propose solutions to the problems of handling the persistent objects of the LHC experiments: event data, calibration data, histograms and so forth. Strong emphasis has been placed on the potential use of standards-conforming, widely-used (commodity) solutions. At an early stage of the project, a potential solution, based upon an Object Database Management Group (ODMG)-compliant Object Database (ODBMS), coupled with a Mass Storage System (MSS) built according to the IEEE Computer Societies Reference Model for Mass Storage Systems, was identified. This potential solution has been the primary focus of our activities, although we have continued to monitor and evaluate alternatives. The preferred components of this solution are built on top of Objectivity/DB and HPSS, coupled with a small quantity of HEP-specific code. This solution has adopted in part (i.e. Objectivity/DB only) or in its entirety by a growing number of experiments at CERN and outside. Objectivity/DB is used for production purposes by several experiments and will, together with HPSS, form the basis of the event storage for many of the experiments that are due to start taking data in or around 1999.

The manpower savings that have been possible by adopting such a solution are already significant. In addition, the functionality provided is much greater than that offered by previous, HEP-specific solutions. The combination of these factors will help us to cope with the much greater volumes of data that we will have to deal with, with increased flexibility, whilst remaining within foreseen man-power constraints.

20 Previous Milestones and Recommendations

We list below the milestones and recommendations from previous reviews of the RD45 project.

20.1 Milestones at the end of the first year (1996)

1. Identify and analyse the impact of using an ODBMS for event data on the Object Model, the physical organisation of the data, coding guidelines and the use of third party class libraries.
2. Investigate and report on ways that Objectivity/DB features for replication, schema evolution and object versions can be used to solve data management problems typical of the HEP environment
3. Make an evaluation of the effectiveness of an ODBMS and MSS as the query and access method for physics analysis. The evaluation should include performance comparisons with PAW and Ntuples.

20.2 Initial Milestones and Recommendations (1995)

RD45 (P59) should be approved for an initial period of one year. The following milestones should be reached by the end of the first year.

1. A requirements specification for the management of persistent objects typical of HEP data together with criteria for evaluating potential implementations.
2. An evaluation of the suitability of ODMG's Object Definition Language for specifying an object model describing HEP event data.
3. Starting from such a model, the development of a prototype using commercial ODBMSs that conform to the ODMG standard. The functionality and performance of the ODBMSs should be evaluated.

It should be noted that the milestones concentrate on event data. Studies or prototypes based on other HEP data should not be excluded, especially if they are valuable to gain experience in the initial months.

21 Glossary

ADAMO - a system, developed in the ALEPH collaboration, based on the Entity-Relationship (ER) model.

ADSM - A storage management product from IBM

AFS - the Andrew (distributed) filesystem

CASE - Computer Aided Software Engineering

CORBA - the Common Object Request Broker Architecture, from the OMG

CORE - Centrally Operated Risc Environment

CWN - Column-wise Ntuple

CTP - Computing Technical Proposal

DFS - the OSF/DCE distributed filesystem, based upon AFS

DMIG - the Data Management Interface Group

EDMS - Engineering Data Management System

GB - 10^9 bytes

HPSS - High Performance Storage System - a high-end mass storage system developed by a consortium consisting of end-user sites and commercial companies

IEEE - the Institute of Electrical and Electronics Engineers

KB - 2^{10} (1024) bytes - normally referred to as 10^3 bytes

LCB - LHC Computing Board

LCRB - LHC Computing Review Board

LIGHT - Life Cycle Global Hypertext

MB - 10^6 bytes

MSS - a Mass Storage System

NFS - the Network Filesystem, developed by Sun

ODBMS - an Object Database Management System

ODMG - the Object Database Management Group, a group of database vendors and users that develop standards of ODBMSs

OID - Object Identifier

OMG - the Object Management Group

OOFS - the Objectivity/DB Open FileSystem

OQL - the Object Query Language defined by the ODMG

ORB - an Object Request Broker

OSM - Open Storage Manager: a commercial MSS

PAW - the Physics Analysis Workstation

PETASERVE - an MSS based upon OSM

PB - 10^{15} bytes

RWN - Row-wise Ntuple

SHORE - Scalable Heterogeneous Object REpository

SQL - Standard Query Language: the language used for issuing queries against databases

SSSWG - the Storage System Standards Working Group

STL - the Standard Template Library: part of the draft C++ standard albeit in a modified form

TB - 10^{12} bytes

TOOLS.H++ - a former de-facto standard container/collection class library, largely made redundant by the collections provided in the standard C++ library

VLDB - Very Large Database

VLM - Very Large Memory

VMLDB - Very Many Large Databases

XBSA - the draft X/Open Backup Services Application Program Interface

22 References

[1] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1998, CERN/LHCC 98-11

[2] Using an Object Database and Mass Storage System for Physics Production [to be completed.]

[3] RD45 Project Execution Plan, 1997-1998, April 1997, CERN/LCB 97-10

[4] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1997, CERN/LHCC 97-6

[5] Object Databases and their Impact on Storage-Related Aspects of HEP Computing, the RD45 collaboration, CERN/LHCC 97-7

[6] Object Database Features and HEP Data Management, the RD45 collaboration, CERN/LHCC 97/8

[7] Using and Object Database and Mass Storage System for Physics Analysis, the RD45 collaboration, CERN/LHCC 97-9

[8] Where are Object Databases Heading? CERN/RD45/1996/4

[9] Why Objectivity/DB? CERN/RD45/1996/6

[10] Objectivity/DB Database Administration Issues. CERN/RD45/1996/7

[11] RD45 - A Persistent Object Manager for HEP, LCRB Status Report, March 1996, CERN/LHCC 96-15

[12] Object Databases and Mass Storage Systems: The Prognosis, the RD45 collaboration, CERN/LHCC 96-17

[13] Object Data Management. R.G.G. Cattell, Addison Wesley, ISBN 0-201-54748-1

[14] DBMS Needs Assessment for Objects, Barry and Associates (release 3)

[15] The Object-Oriented Database System Manifesto M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. In Proceedings of the First International Conference on Deductive and Object-Oriented Databases, pages 223-40, Kyoto, Japan, December 1989. Also appears in [20].

[16] Object Oriented Databases: Technology, Applications and Products. Bindu R. Rao, McGraw Hill, ISBN 0-07-051279-5

[17] Object Databases - The Essentials, Mary E. S. Loomis, Addison Wesley, ISBN 0-201-56341-X

[18] An Evaluation of Object-Oriented Database Developments, Frank Manola, GTE Laboratories Incorporated

[19] Modern Database Systems - The Object Model, Interoperability and Beyond, Won Kim, Addison Wesley, ISBN 0-201-59098-0

[20] Objets et Bases de Données - le SGBD O_2, Michel Adiba, Christine Collet, Hermes, ISBN 2-86601-368-9

[21] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1, 1991.

[22] Object Management Group. Persistent Object Service Specification, Revision 1.0, OMG Document numbers 94-1-1 and 94-10-7.

[23] The Object Database Standard, ODMG-93, Edited by R.G.G.Cattell, ISBN 1-55860-302-6, Morgan Kaufmann.

[24] ADAMO Reference Manual, CERN ECP

[25] HBOOK - Statistical Analysis and Histogramming Package - CERN Program Library Long Writeup, Y250

[26] PAW - the Physics Analysis Workshop - CERN Program Library Long Writeup, Q121

[27] ATLAS Computing Technical Proposal, CERN/LHCC 96-43

[28] CMS Computing Technical Proposal, CERN/LHCC 96-45

[29] HEPDB - A Distributed Database Management System, CERN Program Library Long Writeup Q180

[30] OpenGL Reference Manual, ISBN 0-201-63276-4, Addison Wesley, 1992.

[31] OpenInventor Reference Manual, ISBN 0-201-62491-5, Addison Wesley, 1994.

[32] IRIS Explorer User Guide, ISBN 1-85206-110-3, 1995.

[33] C++ Object Databases - Programming with the ODMG Standard, David Jordan, Addison Wesley, ISBN 0-201-63488-0, 1998

[34] The CMS H2 Object Oriented Reconstruction Project.

[35] The CMS X5 Object Oriented Reconstruction Project, http://cmsdoc.cern.ch/~lucia/tbeam/OO_staff/x5oo.html.

[36] Design Patterns: Elements of Reusable Object-Oriented Software, Gamma, Helm, Johnson, Vlissides, http://consult.cern.ch/books/0201633612.

[37] STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, David R. Musser, Atul Saini, Addison Wesley, ISBN 0-201-63398-1

[38] Object Databases in Practice, Prentice Hall, ISBN 0-13-899725-X.