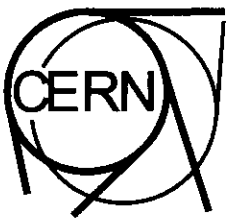

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/LHCC 98-11
LCB Status Report/RD45
6 April, 1998

STATUS REPORT OF THE RD45 PROJECT

The RD45 collaboration
CERN, *Geneva, Switzerland*

This document has been produced for the April 1998 LCB review of the RD45 project. In this paper, we present the status of the project, including a summary of the responses to the milestones[1] set at the 1997 review by the LCB, suggestions for future activities and a revised risk analysis of the current RD45 strategy.

In addition, we describe activities undertaken within various experiments and projects, including NA45, COMPASS, ATLAS, CMS, ALICE, LHCb, BaBar, BELLE and Zeus.

RD45 documents may be obtained through the Web (see http://wwwinfo.cern.ch/asd/cernlib/rd45/index.html) or via e-mail request to the spokesman.

[1] See the minutes of the March 1997 LCB meeting (http://www.cern.ch/Committees/LCB/minutes/mar97/lcb10-3-97.html) for a list of milestones and recommendations.

The RD45 Collaboration

David Malon
Argonne National Laboratory, Argonne, Illinois, USA

Martin Purschke
Brookhaven National Laboratory, USA

Julian Bunn[2], Harvey Newman, Rick Wilkinson
Caltech, USA

Eva Arderiu Ribera, Antoni Baranski, Javier Conde, Dirk Düllmann, Bernardino Ferrero Merlino, Gunter Folger, Marcin Nowak, Jamie Shiers (spokesman)
CERN/IT
Geneva, Switzerland

Pavel Binko, Koen Holtman, Vincenzo Innocente, Arther Schaffer[3], Lucia Silvestris[4], Lassi Tuura, Ian Willers
CERN/EP
Geneva, Switzerland

Martin Gasthuber
DESY/Hamburg, Germany

Andreas Pfeiffer
University of Heidelberg, Germany

David Quarrie
Lawrence Berkeley National Laboratory
Berkeley, CA, USA

Youhei Morita
KEK, Oho,
Tsukuba, Ibaraki, 305 Japan

Stansislaw Jagielski
Faculty of Physics and Nuclear Techniques,
UMM Krakow, Poland

Alexei Klimentov
MIT, USA

---

[2] Permanent address, CERN, Geneva, Switzerland.
[3] Permanent address, LAL, Orsay, France.
[4] Permanent address, INFN, Bari, Italy.

Christian Arnault
Laboratoire de l'Accelerateur Lineaire
Orsay, France

Jacek Becla, Gabriele Cosmo
Stanford Linear Accelerator Center, CA, USA

Sunanda Banerjee
Tata Institute of Fundamental Research
Bombay, India

Simona Rolli, Krzysztof Sliwa
Tufts University, USA

Anwarul Hasan
ETH Zurich, Switzerland and University of Cyprus, Cyprus

Elisa Cargnel
University of Venice, Italy

TABLE OF CONTENTS

1 Executive Summary

Since 1995, the RD45 project has been investigating solutions to the problem of providing persistency to physics data of the LHC experiments, assumed to be in the form of (collections of) *objects*. At the end of the first year, a potential solution, based on standards-conforming products, was presented. During the second year, this possible solution was studied further - performance comparisons with existing systems and tests of functionality and scalability were carried out and production demonstrations were made. The proposed solution is based on a combination of two commercial products, namely Objectivity/DB and HPSS, together with a small amount of HEP-specific code, distributed via LHC++.

In this report, we summarise the activities of the RD45 project during the past year, including progress on the milestones set by the LCB, experience with experiments such as NA45 that have already adopted elements of the solution and other projects such as GEANT-4, together with proposals for future activities.

Objectivity/DB based solutions are being used in production by a growing number of experiments at CERN and outside, and have been selected for a number of high-data volume experiments starting around the year 1999 (BaBar, COMPASS etc.). Based on the results of our research and the adoption of the proposed solution by the community, it is our conclusion, therefore, that the primary goals of the RD45 project have been reached. Further work is clearly required, such as the setting up of full-scale, general purpose, production services. We propose that such services be established in close collaboration with the HPSS-based services offered by IT/PDP group. Associated with such services will be on-going discussions with Objectivity concerning future enhancements, and the provision of HEP-specific class libraries and utilities. In addition, a number of further R&D activities will be required. The precise activities and their respective timescales are currently being discussed with the LHC experiments.

During the past two years, the RD45 collaboration has focused on ODMG-compliant solutions, and has demonstrated the use of a standard, off-the-shelf ODBMS product for storing and managing HEP event data in a production environment.

Despite the focus on ODBMSs, RD45 continues to follow progress in other areas, such as persistent object managers, Object-Relational Databases, including object-oriented offerings from the traditional relational (RDBMS) vendors and so forth.

RD45 continues to participate in the Object Database Management Group (ODMG) - the standards body that defines and maintains the various standards for ODBMSs, as well as the Object Management Group (OMG), and the IEEE Computer Society Executive Committee on Mass Storage Systems (IEEE MSS EC).

1.1 Summary of Activities During Third Year

During the past year the RD45 collaboration has:

- made tests of an Object Database for managing physics data at all stages of production from data acquisition to analysis,
- held several workshops at CERN, focussing on issues related to the RD45 milestones and object databases,
- participated in similar workshops at SLAC and BNL,
- participated in the standards activities of the ODMG and IEEE,
- continued to expose and discuss the proposed strategy with the user community, including at CHEP '97 in Berlin,
- worked with other HEP laboratories, the HPSS consortium and Objectivity to develop a proof-of-concept prototype of the interface between Objectivity/DB and HPSS,
- worked with the GEANT-4, CHORUS, COMPASS, NA45 and NA48 collaborations, in addition to ALICE, ATLAS, CMS and LHCb, on aspects related to object persistency,
- collaborating with experiments at outside laboratories, including ZEUS at DESY, BaBar at SLAC, CDF at Fermilab and PHENIX and STAR at RHIC,
- extended the RD45 test-bed, initially based on a single ODBMS server machine, equipped with 100GB of disk and 2GB of memory, to include a second ODBMS server, equipped with 100GB disk and 1GB memory, which will also act as an HPSS client and test-bed for the Objectivity/DB - HPSS interface,
- made an analysis of the future prospects of various ODBMS vendors,
- investigated the key areas identified by the previous year's risk analysis of the current strategy, namely the possibility of switching to an alternative vendor, database administration issues, and issues related to multi-PB stores.

1.2 Conclusions

The RD45 collaboration has evaluated and compared a number of potential solutions to the problems of providing object persistency services to the event data of the LHC experiments. The preferred solution, based on the use of Objectivity/DB and HPSS, together with a small number of HEP-specific extensions, has been adopted by a number of experiments, both at CERN and at other laboratories. Objectivity/DB - based solution have been used in production by a number of experiments at various laboratories. The next logical step is to offer production services based upon Objectivity/DB and HPSS, and use the experience gained with these services as input to a further round of research and development prior to the final decisions regarding the initial LHC computing models. It is proposed that further HEP-specific developments and enhancements to Objectivity/DB itself be handled as part of the day to day operation of these production services.

2 Overview of Past Activities

The RD45 project has now reached the end of its third year of research. The bulk of its activities have been oriented towards the potential use of standards-conforming, widely-used solutions. The key activities of the past 3 years are listed below.

2.1 Activities of the First Year

During its first year, the RD45 collaboration investigated several different approaches to solving the object persistency problem, including language extensions, persistent object managers and ODMG-compliant ODBMS products.

Using the definition of an ODBMS from the "Object-Oriented Database Manifesto" [15], it was our conclusion that HEP requires a system offering all of the facilities listed as mandatory in this manifesto, all of the features listed as optional, and indeed several others besides!

RD45 was also able to identify an ODBMS product with an architecture offering the required scalability, namely Objectivity/DB and this product has been used for the majority[5] of the prototypes built since, as well as for NA45 physics production runs.

The activities of 1995 can thus be summarised as follows:

- identification of the key standards and technologies involved,
- a clear focus on standard, commercial solutions,
- participation in the appropriate standards bodies,
- extensive training in these key areas,
- evaluation of a number of potential solutions, based upon interim system requirements,
- the identification of a powerful solution to the object persistency problem - a scheme whereby multiple PB of data can be handled without stressing the available technologies,
- close contacts with the appropriate vendors,
- successful prototyping of storage and retrieval (with encouraging performance) of HEP event data,
- wide exposure and general approval of the proposed solution within the community,
- an analysis of the probable capabilities of an Object Database and Mass Storage System [12].

Further details of the activities during the first year can be found in the RD45 status report for 1995 [11].

[5] RD45 continues to monitor possible alternatives, including products from O_2, Object Design International (ObjectStore), POET and Versant. CERN has, or has had, evaluation copies of all of these products.

2.2 Activities of the Second Year

During the second year, the activities of which are described in [4], RD45 continued to focus on ODBMS-based solutions, whilst maintaining a watch on potentially alternative solutions, such as light-weight object managers and object-enhancements to relational databases. The major milestones during the 2nd year were:

- the demonstration of analysis using an ODBMS-based solution with performance equivalent to that of today's solutions [5],
- an evaluation of the functionality of ODBMS features such as schema evolution, object versioning and data replication and their applicability to HEP [6],
- an analysis of the impact of ODBMS features on the object model, the physical organisation of data, coding guidelines and the use of 3rd party class libraries [7].

2.3 Activities of the Third Year

During the past year, the RD45 collaboration has concentrated on demonstrating the feasibility of using an ODBMS-based solution for object persistency in typical production scenarios. This includes the complete chain from data acquisition or simulation through to end-user analysis. Although the data volumes involved have clearly not been at the level anticipated at the LHC, the basic functionality has been shown to work successfully in a production environment.

In addition to these activities, there have been a significant number of changes that are worthy of note. Most importantly, several key experiments that are due to take data in 1999 and beyond, have formally decided to use Objectivity/DB (and HPSS) as key elements of their offline strategy. These experiments include COMPASS and NA45 at CERN, BaBar at SLAC, and the RHIC experiments at Brookhaven. All of these experiments will take very significant quantities of data - typically in the 100-500TB/year range - at data rates of around 35MB/second. In other words, they present a challenge at least as significant as that posed by the LHC experiments - similar data volumes and data rates, but much earlier in time. In other words, they will not be able to benefit from the evolution in technology that one can expect between now and the startup of the LHC, but will have to develop solutions based on today's technology. Clearly, the lessons learnt from these experiments will be of great relevance to the preparation for the LHC.

In addition to these "future" experiments, there is a ever growing number of existing experiments that have already deployed production applications on Objectivity/DB. Today, this list includes AMS, CHORUS, NA45, NA48, OPAL and ZEUS, with more expected. In particular, the recently proposed LEP data archive project current plan to store some tens of TB of LEP data in the assumed "LHC solution", namely Objectivity/DB. At least two of the LEP experiments, ALEPH and DELPHI, have already started projects to investigate the use of Objectivity/DB and the other LHC++ components for current analysis.

2.4 Conclusions

Hence, it is our conclusion that the major goals of the RD45 project have been achieved. A significant amount of work clearly remains - not least, the delivery of the main outstanding enhancement requests and the setting up of full production services. In addition, one would expect many new and exciting developments in the areas of optimisation, distribution, parallelism and so forth. These topics, very worthy of detailed research, could form the basis of a follow-on project, but maybe best addressed in the light of the initial experience of experiments such as BaBar and COMPASS.

3 Milestones from the March 1997 LCB Review

The RD45 project was reviewed by the LCB in March 1997, and recommended for continuation for a further year, with the following milestones and comments:

"The RD45 project has continued to make excellent progress in identifying and applying solutions for object persistence for HEP and the LHC collaborations have shown great interest in their work.

RD45 has successfully addressed the milestones set by the LCRB for 1996 and the LCB recommends that the project be approved for a further year during which the details of the work-plan should be defined in conjunction with the LHC collaborations.

The LCB agrees with the program of work outlined in the RD45 status report and sets the following milestones for the third year of the project:"

1. Demonstrate that an ODBMS can satisfy the requirements of typical simulation, reconstruction and analysis scenarios with data volumes of up to 1TB.

2. Investigate the impact on the every-day work of the end-user physicist when using an ODBMS for event data storage. The work should address issues related to individual developers' schema and collections for simulation, reconstruction and analysis.

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

In addition, the project is asked to include the following activities in its work-plan:

- Participate in the US-based related projects to gain experience in wide-area replication and use of a MSS.
- Investigate issues concerning porting data and applications from Objectivity to an alternative vendor's ODBMS product.
- Extend the performance comparisons made for the previous year's third milestone to investigate the potential advantages of adopting a new ODBMS based approach as the query and access method for physics analysis.
- Measure rates at which Objectivity can be populated with the aim of simulating event data input from a data acquisition system.
- Provide a guide to ODBMS usage for non-expert physicists and explain how it impacts their work.
- Provide guidance to LHC collaborations in the development of related database applications, such as management of calibration data etc.

Work on these milestones and recommendations is covered in detail below.

4 The Project Execution Plan (PEP)

At the request of the project referees, a detailed project execution plan [3] for the current year was prepared. This PEP describes the scope of the project, the work-plan, methodology, assumptions, detailed sub-tasks and resources. We repeat below the proposed schedule that was established, together with the actual completion date of the various tasks.

TASK	SCHEDULED COMPLETION DATE	ACTUAL COMPLETION DATE	COMMENTS
Install Objectivity/DB V4.0.2	April 1997	April	Installed in LHC++ tree in AFS
Test Replication (DRO) across heterogeneous platforms in LAN	May 1997	April	
Setup problem tracking system for bug reports and requirements	May 1997	June - a common system was setup for all LHC++ components	Based on GNATS
Joint meeting with HPSS consortium and Objectivity to define implementation schedule for Objy/HPSS interface	May 1997	May + follow up meetings in September and November. Proof of concept prototype delivered prior to SuperComputing 97.	Including representatives from other interested HEP laboratories (Caltech, LBL, SLAC etc.)
Participate in Objectivity/DB user meeting	May 1997	May	
Test schema evolution in scenarios typical of HEP	June 1997	July	
Feed HEP requirements into ODMG meeting	July 1997	July	Set post-V2.0 directions
Test WAN DRO	July 1997	July	Requires installation of DRO at remote sites plus local Objectivity/DB expertise
RD45 workshop on HepODBMS	July 1997	July	Define contents/schedule of alpha release. 2 day workshop including working sessions
PEP regarding CMS CDR project	July 1997	December	ATLAS & CMS PEPs presented to December LCB
Participate in Kyushu field test	August 1997	July - October	Assumed that Kyushu field test begins in May 1997. Start delayed until July.
Investigate porting of applications from Objectivity/DB to Versant	September 1997 (V5.0 release date + 2 months)	Delayed one month due to unavailability of Versant consultant	Assumes Versant V5.0 is released in July/August 1997
PEP regarding ATLAS project(s)	September 1997	December	See above
Alpha release of HepODBMS	October 1997	Alpha releases made in July and November (LHC++ roadshow)	

Perform performance and scalability tests with Versant	November 1997	Versant training postponed until October. First results presented in December	Versant V5.0 release + 4 months
Test the use of multiple federations and offline partitions for handling (semi-) private schema and data	December 1997	July	Initial results already at July workshop
Participate in Lassen field test	December 1997	Lassen merged with V5 - beta did not take place.	Assumes that Lassen field test begins in September 1997
Deliver prototype meta-data browsers and search engines	December 1997	Available in Java	
Demonstration of prototype Objy/HPSS interface	December 1997	November 1997, IBM only	Restricted to certain platforms, e.g. IBM?
Beta release of HepODBMS	March 1998		
Meet LCB milestones	March 1998		Requires appropriate input from experiments

Table 1 - detailed project execution schedule

4.1 ATLAS Addendum

As indicated in the above table, an addendum to the RD45 PEP was produced, covering the milestones related specifically to activities within the ATLAS collaboration. These milestones are given below.

Milestone	Description	Date
1	Geant-3 digits for all detector systems in ODBMS	January 1998
2	first version of a detector description database available This should be interfaced to the current simulation and reconstruction as well as ATLAS Geant-simulation and OO reconstruction	June 1998
3	Geant-4 event data available for parts of the ATLAS detector	December 1998

Table 2 - Summary of ATLAS milestones for ODBMS-related work

4.2 CMS Addendum

Milestone	Description	Date
Data retrieval	Prototype reclustering algorithm based on HepODBMS	June 1998
As above	Prototype of strategies for data organisation and access	December 1998
Event data model, reconstruction and analysis framework	Prototype of a reconstruction framework for test beam and simulated data	June 1998
As above	Release of a "User analysis environment"	December 1998
Calibration database prototype	Integration of the present prototype with Objectivity Version 5	March 1998
As above	Release of a class library and documentation	June 1998

Table 3 - Summary of CMS milestones for ODBMS-related work

5 Risk Analysis

5.1 Introduction

The main risks of the RD45 strategy were analysed as part of the work reported on at the March 1997 review and are described in the associated status report [4]. The main risk factors identified at that time are listed below.

5.2 Key Risk Factors

1. The Mass Storage System - only HPSS appears to be capable of scaling to the multi-PB region and is currently only available on IBM[6] platforms. It is not yet in production with large volumes[7] of data, nor is it likely that it will ever become a "commodity" product, i.e. with more than 100,000 deployed licenses. On the other hand, all major US projects plan to use HPSS, and it is therefore likely that a solution capable of meeting at least the requirements of these projects will exist, even if this is not a commodity item and implies the use of a restricted set of hardware (today at least),
2. The deployment and administration of a fully distributed ODBMS (+MSS) in the wide area. The issues involved are not fully understood, and careful investigation into the risks and manpower involved needs to be made. However, there are a number of widely distributed ODBMS projects and hence this issue will need to be addressed by the industry,
3. The survival of a given ODBMS vendor. Although the ODMG standard promises portability of both code and data between the products of different vendors, it is not clear if a product other than Objectivity/DB offers sufficient scalability as to offer a realistic alternative. However, as Objectivity/DB and other ODBMS products [38] sell strongly into the Telecoms and financial markets, which are both healthy and assured of survival, products capable of meeting the requirements of these markets can be expected to continue to exist. The evaluation of the preferred alternative solution is described in section 6.1.4 on page 25. The overall architecture of Objectivity/DB and its associated performance and scalability are not considered to be risks. Even if no enhancements were made to the product over what is available in the latest release then available, V4.0.2, it would be possible, with a few minor restrictions, to use this system for LHC production. Similarly, the availability of appropriate hardware on which to base the system is not considered to be a risk factor.

[6] This is still true, although porting efforts are in progress at a number of laboratories, including Caltech (HP/UX), CERN (Digital Unix) and SLAC (Sun).
[7] i.e. >> 1TB. At SuperComputing '97, site summaries suggest that the average data volume is still well below 10TB.

5.3 Conclusions from Risk Analysis

The above risk analysis suggest that there are three main areas of concern, which need to be addressed. These are:

1. The availability of a commodity, but high-end, mass storage system,
2. Issues related to wide-scale deployment of a fully distributed ODBMS,
3. The availability of a suitable, scalable, object database.

We consider the provision of a suitable MSS to be outside the mandate of the RD45 project and rely on an appropriate system being identified, acquired and run by IT/PDP group. Coordination between IT/PDP group and the RD45 project is ensured by regular meetings, which currently focus on Objectivity/DB - HPSS issues. Nevertheless, it is worth pointing out that the mass storage area continues to evolve. As was discussed at CHEP '97 in Berlin, HPSS is not the only MSS to be successfully deployed in HEP. A number of other sites make production use of the Lachman/Legent/Computer Associates Open Storage Manager (OSM) product, although there are concerns as to whether this will scale to the PB region. SGI is also known to be working on a high-end MSS, which presumably re-uses some components from the Cray Data Migration Facility. Most important of all, however, are analyses that suggest that PB stores will simply be common-place in the year 2005.

To address these issues, we have built a number of database administration (DBA) tools (see section 10.3 on page 84) and are studying closely the issues involved in an eventual migration from the product of one vendor to another, including an evaluation of the currently preferred fall-back solution. These topics are covered in more detail below.

5.4 The Use of Commodity Solutions

The adoption of commodity solutions is perhaps the best protection against the risks that are described above. As an example, we cite the case of a Web browser.

A Web browser is clearly a tool that is expected on every machine; this is not limited to the HEP community, but is valid world-wide. As such, we can be confident that such functionality will be provided, perhaps in a different guise, in the future. Whether a particular browser dominates the market is, personal preferences aside, irrelevant. The feature set of the different browsers is now so similar that there is little to choose between them.

Clearly, the same cannot yet be said of object databases - they are perhaps in the situation that browsers were in prior to release 3 of Internet Explorer. However, the ODBMS market is continuing to grow, and includes segments such as the telecoms industry, which provides the demand for performant, distributed solutions. Areas such as digital libraries will create the demand for highly scalable solutions. Finally, their match to C++ and Java - and the Web itself - should help to maintain demand.

6 Work on Referees' Recommendations

In addition to the milestones set for the past year, the RD45 project was asked to:

- Gain experience with wide-area data replication and the Objectivity/DB mass storage interface,
- Investigate porting applications and data from Objectivity/DB to an alternative ODBMS,
- Extend the performance comparisons made as part of the previous year's activities to explore the potential advantages of adopted an ODBMS-based approach for physics and analysis,
- Provide a novice guide to ODBMS usage for end-users,
- Provide guidance to the LHC collaborations in the development of ODBMS-based applications, such as for calibration data.

These issues are covered in more detail below.

6.1 Data Replication

During the second year of RD45, detailed studies of the data replication capabilities of Objectivity/DB were made. However, these tests, which are described in detail in [6], were performed with pre-release software on a single platform (NT), and were restricted to LAN tests. We have since extended these tests to include large numbers of heterogenous hosts both in the LAN and WAN. We report on these results below, together with an analysis of how replication, and other data distribution tools in Objectivity/DB, could be used to solve data distribution scenarios typically of HEP.

User-data replication has been supported in Objectivity/DB since release 4.0 in early 1997. Replication works at the level of a physical database - that is, a database is either replicated or not, each replica being termed an *image*. To use the replication feature, one must first partition the federation into a number of *autonomous partitions*, using the Objectivity/DB Fault Tolerant Option (FTO). Each partition has its own lock server, replica of the system information (schema and catalog) as well as one or more database servers and associated databases. For example, one could imagine that CERN and each regional centre would operate as separate partitions. In such a scenario, each database could be replicated to one or more regional centres.

A full description of the replication option of Objectivity/DB is provided in [6] and will not, for space reasons, be repeated here.

6.1.1 Tests Across Heterogeneous Platforms

Since the official release of Objectivity/DB version 4, user data replication has been available across all supported platforms. It has been tested across 7 different operating system/platform combinations, namely Digital Unix, HP/UX, IBM, SGI, Solaris and NT running on both Alpha and Intel architectures. These tests, which were largely confined to the local area, functioned correctly.

6.1.2 Tests With Large Numbers of Images

In certain scenarios, such as the replication of calibration or tag data, it is likely that large numbers of replicas (images) would be involved. Comparisons with HEPDB [29] suggest that some 10-15 images might be required for calibration data, but many more for tag data - perhaps as many as the number of institutes involved in an LHC collaboration. We have therefore tested replication up to 100 images - the limit arriving from the number of nodes that could conveniently be used for this purpose and not from any limitation in Objectivity/DB.

As shown in the figure on page 24, the time taken to both create persistent objects and commit the corresponding transaction increases with the number of images involved. The latter is expected - not only does the transaction not complete until the data involved has been safely written to disk on all servers, but more network traffic is involved. By default, Objectivity/DB will send the data to 4 servers in parallel, although this may be increased by a configuration parameter. When an attempt to create new objects is made, the database will dynamically contact all servers involved and only permit the operation to continue if sufficient *quorum* is obtained. This technique, similar to that deployed in VMSClusters, ensures that database consistency is maintained.

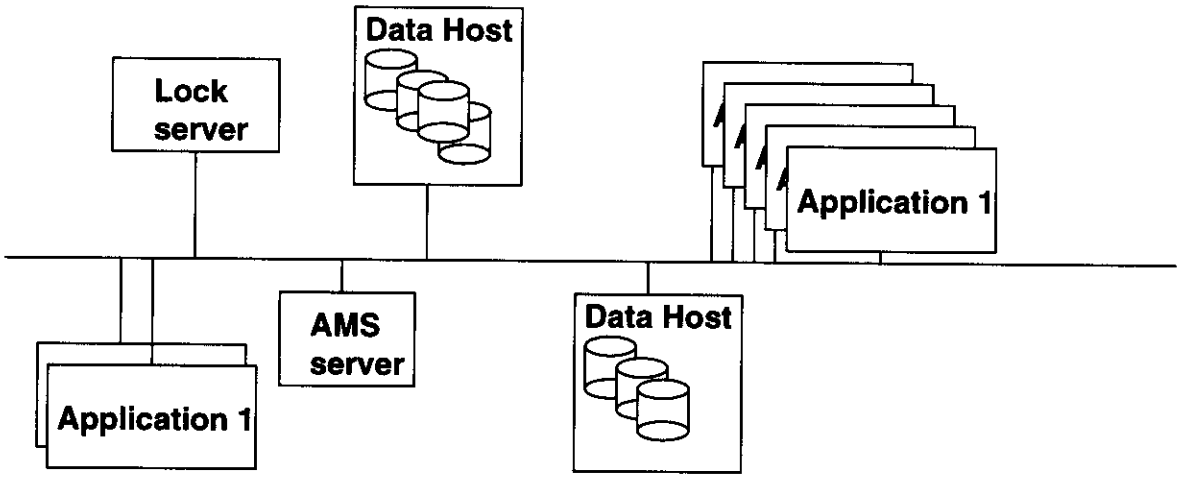

Figure 1 - An Objectivity/DB Federation with a Single Partition

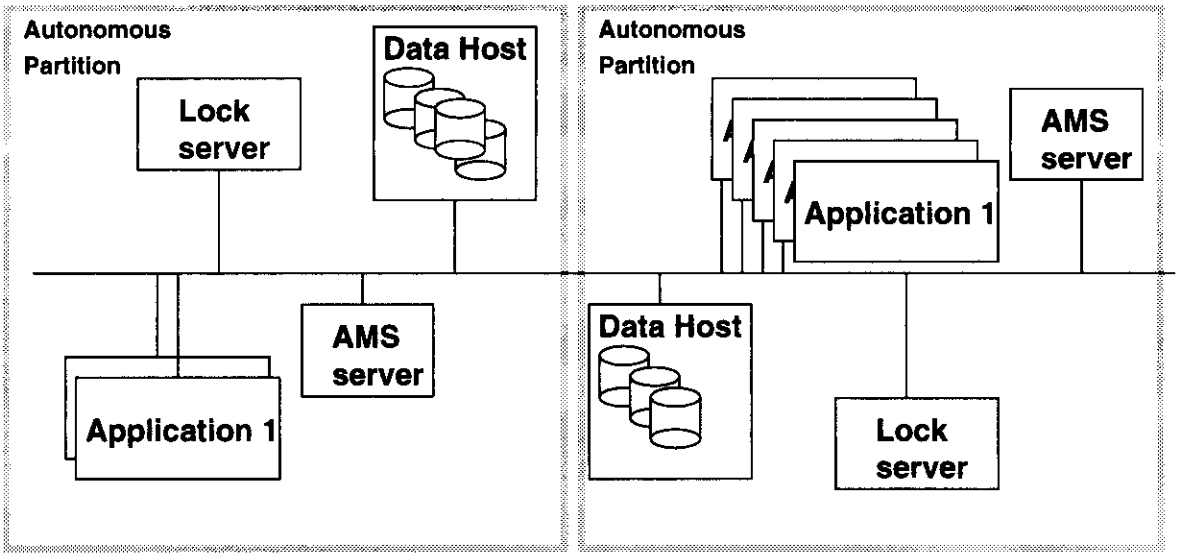

Figure 2 - An Objectivity/DB Federation with 2 Partitions

6.1.3 Wide-Area Tests

The tests described above where initially made using just systems at CERN. They were later extended to include machines in the wide-area, including nodes in Caltech. In these tests just 3 images were involved: two at CERN and the third in Caltech. The tests, made to simulate the update frequency and data volume of a calibration database, involved updating 1KB of data every 5 minutes. As the following figure shows, the data rate was strongly correlated to the hour of the day. During peak hours, when the link is essentially saturated, a relatively low data rate of around 2Kbit/second was achieved. However, during off-peak hours, data rates of 20Kbit/second were observed. Under such conditions, remote replicas behave essentially identically to local ones.

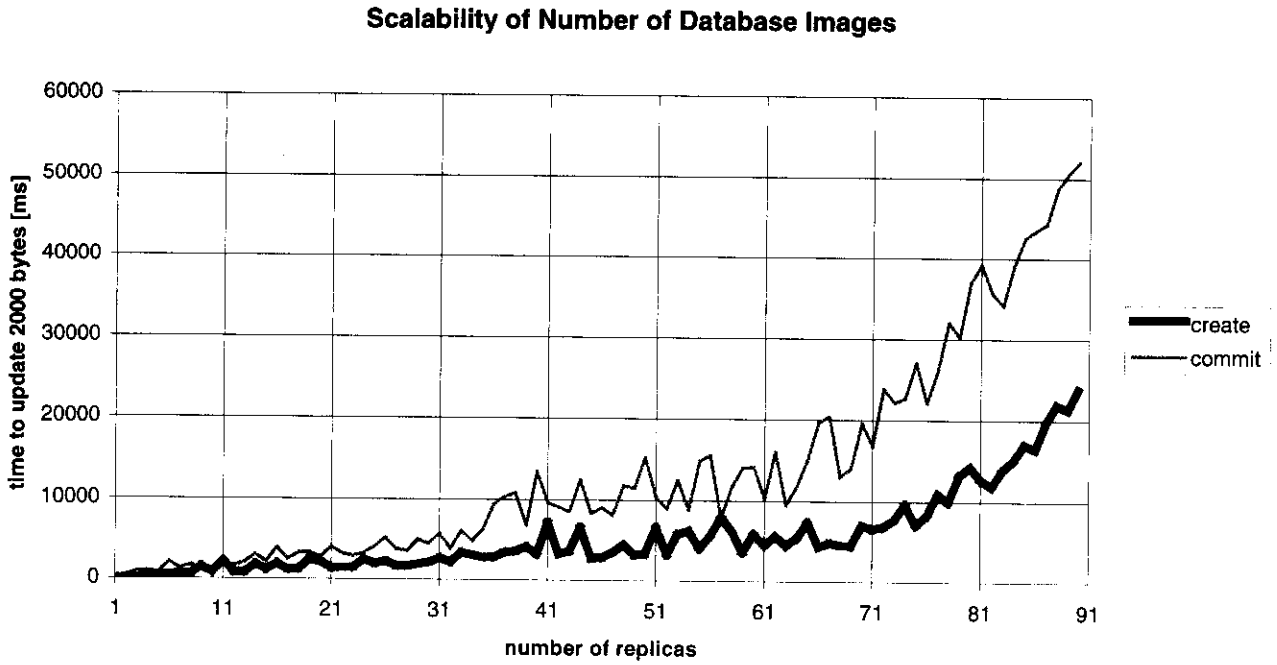

Figure 3 - Replication using Large Numbers of Images

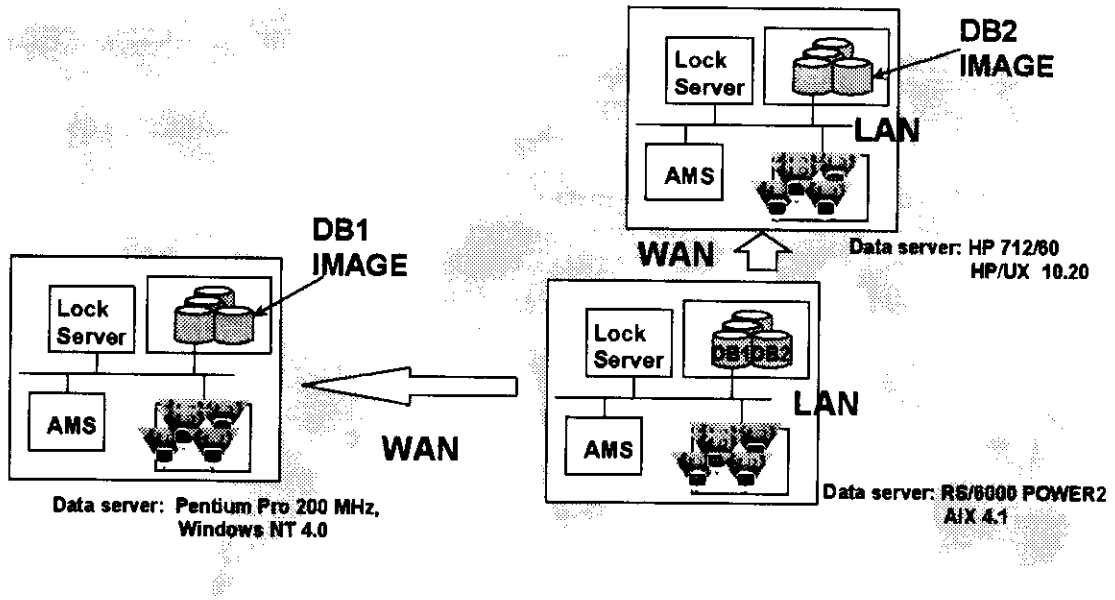

Figure 4 - Wide-area Test Configuration

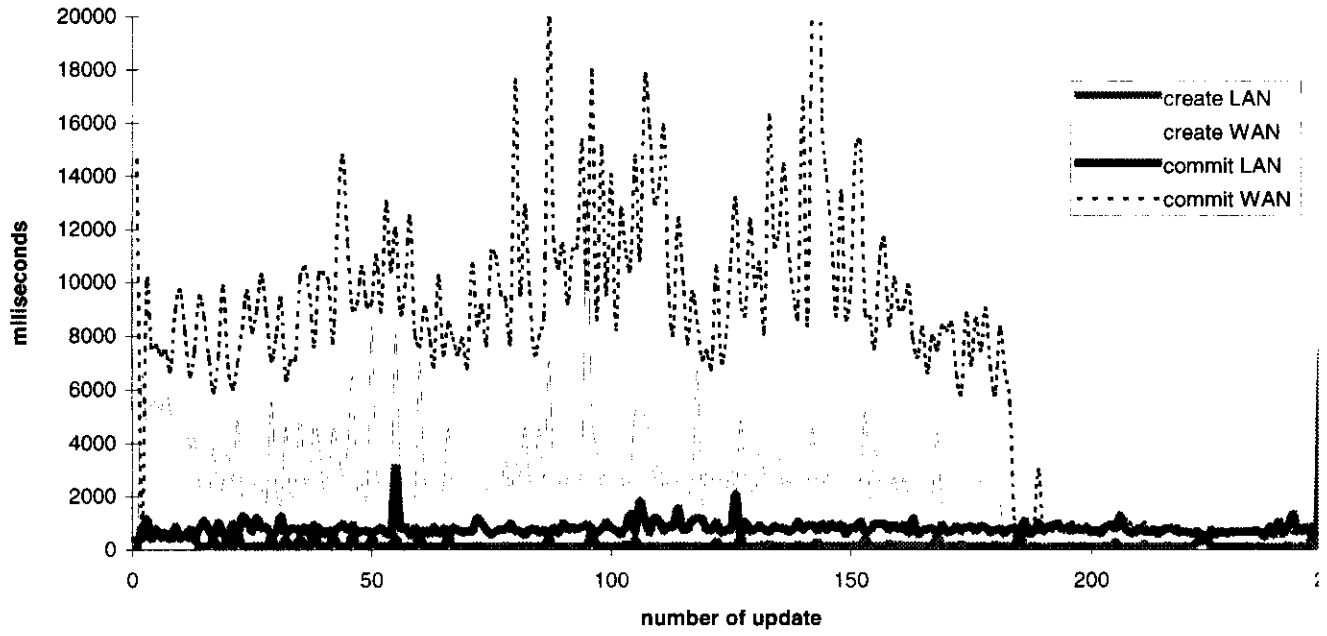

Figure 5 - Tests of Wide-Area Data Replication

6.1.4 Enhancement Requests

A number of the enhancement requests made by CERN have already been addressed by Objectivity. For example, improvements have been made in the area of software control of offline images and in the re-sychronisation of images. Other enhancement requests remain pending. These include the ability to replicate complete databases by offline media, i.e. tape, rather than the network, and to provide more flexibility in the selection of which image is used - currently the product differentiates only between local and remote, but has no concept of nearest or least loaded server. In addition, improvements have been requested to the underlying Fault Tolerant Option (FTO). In the current version, catalog changes, such as modifications or additions to the schema or the addition of new databases, are only possible if all partitions are online. We have requested that the same quorum mechanism be utilised to permit updates to "system" data even if one or more partitions are unavailable, as is currently the case with user data. Appropriate enhancements will be made to a future version of Objectivity/DB.

6.1.5 Conclusions on Replication

We have verified that the basic functionality offered by the Data Replication Option (DRO) of Objectivity/DB behaves as documented. A number of bugs exist in the version that was tested (4.0.2) which are scheduled to be fixed in the production release of version 5. We believe that, once the outstanding enhancement requests have been satisfied and we have been able to verify that the outstanding bugs have been corrected, this functionality satisfies our key requirements for data distribution. However, it is important to stress that the required network bandwidth must be made available - it is not realistic to replicate large data volumes, e.g. in the TB range, over the networks that are typically in use in HEP

today. Thus, in the short term, "offline replication" remains the most appropriate option for large data volumes. Replication remains a viable solution, however, for smallish data volumes, such as in the case of calibration data.

6.2 Mass Storage Interface

As described in the March 1997 status report [4], an interface between Objectivity/DB and HPSS at the level of the Objectivity/DB server is the preferred method of coupling these two systems. The Objectivity/DB server, which deals with database pages and has no knowledge of objects, uses standard I/O calls to read/write data from databases - exposed to the operating system as files. This matches well to the HPSS client API, which offers a similar interface: I/O calls such as *open(), read(), seek()*, etc. have equivalents such as *hpss_open(), hpss_read(), hpss_seek()*. A series of meetings between representatives of the HEP laboratories planning to use both HPSS and Objectivity/DB and Objectivity themselves resulted in an agreed timescale for the delivery of a production interface between the two products. The basic milestones were:

- Proof-of-concept prototype, meeting basic requirements, to be demonstrated by the time of SuperComputing '97 (November 1997),
- A full production version, no later than the end of 1998.

The requirements for both prototype and full production versions are listed below.

The actual interface has been designed such that an alternative Mass Storage System can be used, if required. The interface layer has been defined - Objectivity will deliver a linkable version of their server, which can then be interfaced at the client site to the MSS of choice. This also allows for additional monitoring code, or for example access control checks, to be inserted at the level of database (file) open/close. The proof-of-concept prototype - currently limited to IBM platforms only (the only platform on which the HPSS client is currently supported, although ports to Sun and DEC are under way) will be stress-tested at SLAC, CERN and other laboratories throughout 1998.

6.2.1 Requirements for Proof-of-Concept Prototype

1. The prototype should demonstrate client-transparent migration/staging of databases to/from tertiary storage, i.e. the disk pool should be smaller than total federation
2. A demonstration of basic database functionality, such as sequential and random read and write access, the creation and deletion of databases, is required.
3. Heterogeneous client access should be made, i.e. access from both Unix and NT clients.

6.2.2 Requirements for Production Version of Objectivity/DB - HPSS Interface

1. A full-production version of the Objectivity/DB - HPSS interface must be provided by Q4 1998 at the latest.

Justification: CERN plans to use this interface during 1999 for production with some 300-400TB of data for two experiments: COMPASS and NA45. In addition, we anticipate pre-production usage by several other experiments, including the LHC experiments themselves. Other HEP laboratories, such as SLAC, also have plans for full-scale production in the same time frame.

2. The production version of the Objectivity/DB - HPSS interface must be supported on the main platforms in use in the HEP community. For the first production release, the limiting factor is likely to be the supported platforms for the HPSS client API and disk mover. The bare minimum list is IBM (CERN) and Sun (SLAC). Other platforms, such as HP and DEC, for which HPSS support is envisaged on this timescale, should also be added if possible.

   Justification: the choice of data management software should not restrict customers to one or even a few hardware vendors. It is essential that we are able to obtain commodity hardware offering the best price/performance at any given time.

3. Alpha and Beta-releases of the Objectivity/DB - HPSS interface must be made available as early as possible for stress testing by the HEP community. Ideally, the proof-of-concept prototype should be made available at the time of SC97 or earlier, and an extended beta-test should be planned starting in June 1998.

   Justification: Extensive data-intensive testing will be required to ensure that the product is production-ready prior to committing many hundreds of TB of data to the system.

4. The Objectivity/DB - HPSS interface must be part of the standard Objectivity/DB product line, and not a "special one-off" version.

   Justification: HEP experiments plan to use this software for up to 25 years! The longevity of the software must be guaranteed.

5. No source-code changes to Objectivity client applications should be needed to access Objectivity/DB databases that are (partially) stored in HPSS-managed storage.

   Justification: The operation of the modified AMS server should be totally transparent to Objectivity client applications, apart from timing and throughput. Specification of the federated database configuration and location should be through a normal Objectivity boot file.

6. The AMS server should be able to access tape-resident databases with no explicit overriding of timeouts in the Objectivity client code.

   Justification: The Objectivity client should be unaware of the location of databases in a distributed environment. Accommodating the possibly long time-scales involved in HPSS operation should not adversely impact the timeouts and hence security of non-HPSS operations.

7. It must be possible to construct an Objectivity/DB federation where some data is stored in non-HPSS managed storage and some in HPSS. A solution whereby such databases were separated on different database servers, managed by separate AMSs, would be an acceptable solution.

   Justification: Some fraction of the federation, such as meta-data, should never be migrated off disk, as it will be required for fast access to other parts of the data. In addition, some databases may reside on private workstations/PCs, or be distributed in the WAN. A solution that required all data to be managed by one or more HPSS-instances, and hence ruled out e.g. databases/partitions on the desk-top, would not be acceptable.

8. Federations created on top of the Objectivity/DB - HPSS interface should provide all the functionality of non-HPSS stored federations, including replication, partitions etc.. In addition, the various tools such as *oonewdb*, *oochangedb* etc. should work as normal.

   Justification: The purpose of providing the interface to HPSS is to increase the size of federations that are possible. No reduction of functionality should occur.

9. Access to data stored in disk-resident databases in HPSS-managed storage should be at least as fast as to non-HPSS managed storage, hardware and other considerations not withstanding.

   Justification: The Objectivity/DB - HPSS interface should not introduce bottlenecks or adversely affect performance.

10. The Objectivity/DB - HPSS interface must provide support for federations well in excess of 1PB.

    Justification: Objectivity/DB will be used together with HPSS to store many tens of PB of data - some 5PB/year for the LHC experiments. A demonstration of a 1TB federation is an acceptance test for the interface.

11. The Objectivity/DB - HPSS interface must support the basic throughput and load requirements of the LHC experiments. Basically, these call for:

    - 100MB/second per logical stream for ATLAS/CMS data acquisition,
    - 1.5GB/second per logical stream for ALICE,
    - Some 100 streams reading/writing at a few MB/second for data analysis, reconstruction and reclustering,
    - A total of some 150 concurrent users connected to the federation performing analysis.

Although the system configuration will clearly have a strong impact on the feasibility of satisfying these requirements, the actual interface between Objectivity/DB and HPSS should be written in such a way as to minimise overhead. A possible example would be the use of the "advanced" HPSS API, rather than the basic API, which avoids routing all I/O calls through the HPSS server.

12. It must be possible to force Objectivity/DB to access data via the AMS, even if the data reside locally.

    Justification: access to HPSS-resident data will only be supported via the AMS. To reduce network load, these data are expected to be stored on the same machine as the AMS.

13. A means of passing "hints" and other information to HPSS must be provided.

Justification: certain characteristics for HPSS-resident data should or must be specified at file-creation time. In addition, to optimise performance, e.g. to allow the server to pre-fetch data blocks, a mechanism must be provided whereby such information, known only to the client, can be passed to the server.

6.2.3 Conclusions on MSS Interface

The need for an interface between Objectivity/DB and HPSS has been identified. Agreement has been reached between members of the HEP community, Objectivity and the HPSS consortium as to how such an interface could be provided, namely via interfacing the Objectivity/DB server to the HPSS client API. A prototype of such an interface has been developed and is currently under test at a number of HEP institutes, including Caltech, CERN and SLAC. A production version of the interface - with enhancements as identified during the test period - is scheduled for deliver by the end of 1998. Tests of this prototype interface are described in section 9 on page 62.

6.3 Evaluation of an Alternative ODBMS

One of the largest challenges facing today's very long-lived HEP experiments is that of adapting to the highly dynamic computing environment. The lifetime of a typical LHC experiment is very long compared to the timescales of the computing industry. This is clearly demonstrated by looking back some 20-25 years. During this period, we have seen dramatic changes, and changes no less significant must be expected between now and the end of LHC data taking. At CERN, we have seen the introduction of general-purpose networks, of interactive computing and of distributed computing. However, collaborations were largely able to choose a computing environment that remained relatively stable throughout the entire lifetime of the experiment. This was no longer true with LEP, which had to face major changes such as the migration from CERNVM to Unix.

Thus, irrespective of issues relating to the long-term survival of any hardware or software vendor, it is essential to be prepared for change. As such, we have investigated the issues related to migrating between the products of two ODBMS vendors, namely Objectivity/DB and Versant. This evaluation comprised several steps:

- An investigation of the scalability and limits of the alternative architecture,
- An investigation of the effect of the programming interface on HEP applications,
- An investigation of the impact on the ODBMS architecture on HEP data models.

6.3.1 Scalability Issues

In Versant, a distributed database may consist of up to 2^{16} databases. Each of these databases may consist of up to 2^{16} volumes, which in turn map to files. The primary, or system, volume, contains the schema for that database - as noted below, there is no built-in mechanism whereby consistent schema across the set of databases can be ensured. The current version of Versant appears to be limited to 2GB/volume. However, a more likely limit is the available disk space on a given server, assuming that all volumes belonging to a given database are stored on the same server.

Tests have been made up to 35GB databases and up to 1300 volumes per database. Using multiple databases, a distributed database of roughly 0.5TB has been created.

Limit	Objectivity/DB	Versant	Comments
Object ID	64 bits	64 bits	
# of Databases	2^{16} - 1	2^{16}	
Maximum DB size	Filesystem limit	N/A	Versant maps volumes to files
# containers/DB	2^{15} - 1	2^{16}	"volumes" in Versant (≤ 2GB)
# pages/container	2^{16} - 1	2^{16}	
Page size	≤ 64K	16K (fixed)	

Figure 6 - Limits of Objectivity/DB and Versant Architectures

6.3.2 Architectural Issues

Versant is built upon a very different architecture to Objectivity/DB. For example, whereas Objectivity/DB implements a "fat-client/thin-server" model, Versant provides the opposite. Again, whereas the object identifier (OID) in Objectivity/DB has a direct physical mapping, Versant has a logical object identifier (LOID). From the HEP point of view, we believe the most important issues to be the following:

- In Versant, distributed databases are much less tightly coupled than in Objectivity/DB. As such, each database has its own schema, as opposed to Objectivity/DB, where the schema are shared across the entire federation. In our environment, where many databases are likely to reside offline, and hence have inaccessible schema, inconsistencies between the schema of different databases are bound to happen.
- Although Versant implements a LOID, an application must know in which database *each* object was created and is responsible for opening the databases in question. This contrasts strongly with Objectivity/DB, where it is sufficient to initialise access to the federation,
- Object clustering is much more primitive in Versant than in Objectivity/DB.

Of course, there are areas, such as the provision of access control, where Versant is superior to Objectivity/DB. However, the issue is not so much to compare the features of the two products, which clearly must be done in the light of the specific requirements of the applications that will be built on top. Rather, it is to understand whether Versant can be considered a viable *alternative*, e.g. as the basis of a fall-back strategy. It is clear that these products - and the other ODMG-compliant offerings - are not different implementations of an identical specification. Many features are common, but others, such as the above-mentioned clustering strategies, differ significantly.

6.3.3 Performance Comparisons

A number of basic performance measurements were performed using Versant and compared with Objectivity/DB. In particular, the read and write performance with varying object size was measured. Both products show very similar dependencies on the database page size, with a dip in performance around the page size. The tests were done with local data which, in the case of Objectivity/DB, results in direct I/O from the client to the database. Hence, although the results show consistently better performance in the case of Objectivity/DB, we would expect the advantage to be less in the case of remote data. On the other hand, Versant's finer grain locking mechanism - object rather than container level - results in more lock traffic and hence reduced performance.

6.3.4 Porting Issues

A number of relatively small applications, such as the RD45 limits and scalability tests and the Caltech "stars" benchmark, have been reimplemented and/or ported to Versant. Whilst these activities have confirmed that it is indeed feasible to move applications, the

difficulties in moving a large user community and/or large data volumes should not be underestimated.

Given the architectural differences described above, class libraries such as HepODBMS, which implement clustering strategies, would have to be redesigned to work with Versant, rather than Objectivity/DB. There would also be implications for applications such as HepExplorer, as the various modules would have to explicitly open the required databases and not simply connect to the federation. Moreover, the issue of data conversion between the two systems would need to be studied. We believe that these issues would be best addressed by taking a "non-trivial" application and moving both the data and code to a different ODBMS. A possible example of such an application and data could be the CMS H2 test-beam analysis applications. The porting of these applications would also require the conversion of some 80GB of associated data, currently spread over more than 1000 databases. However, the number of classes involved (of the order of 10) is significantly lower than expected for the full object model of an LHC experiment, where of the order of 1000 classes are anticipated.

6.3.5 Conclusions on an Alternative ODBMS

We believe that the performance and scalability measurements of Versant confirm our primary choice of Objectivity/DB, whilst nevertheless demonstrating that Versant is a possible alternative should, for example, Objectivity/DB cease to exist. However, it should be stressed that migrating from one solution to another is a significant undertaking that is likely to have implications on the physical location of data, if not also the logical object model.

6.4 ODBMS-based Data Analysis

In the context of LHC++, a data analysis framework based on industry-standard tools, together with HEP-extensions, has been built up. These tools, which have been demonstrated in a series of roadshows at CERN and outside, are largely based on the use of IRIS Explorer [32], together with HEP-specific extensions (see chapter 10 on page 82.) A more detailed description of ODBMS-based data analysis is given in the discussion on milestone 2 (see section 8 on page 47).

6.5 Novice Guide for End Users

An introduction to the overall LHC++ framework has been prepared as part of the initial LHC++ roadshows, which concentrated on end-user tasks such as histogramming and data analysis. This guide, available in both printed form through the CERN Program Library office and via the LHC++ Web pages, covers all aspects of Objectivity/DB that need to be exposed to end users. In fact, as this guide was prepared prior to the setting up of production servers - scheduled for the first half of 1998 - some additional complications, related to the use of Objectivity/DB with AFS have had to be exposed. In the medium term, these issues will be greatly simplified and users will find that the default environment is

such that they can use Objectivity/DB-based applications with no additional setup. However, a general knowledge of some of the ODBMS terms[8] is likely to be useful - just as an overview of the basic ZEBRA terminology (banks, stores, divisions etc.) was in the past.

Clearly, the use of an Objectivity/DB federated database for the storage of physics data, calibration data and histograms has an impact on the environment that users see. Notwithstanding the differences in functionality, it is perhaps useful to compare some key features of the new environment with that required for existing CERNLIB packages, such as FATMEN and/or HEPDB.

In the case of Objectivity/DB, an environment variable, *OO_FD_BOOT* is used to locate a so-called *boot-file*, which contains information on the federated database, such as the *lockserver host*, the location for journal files and the federated database catalogue. This is very similar to HEPDB: here, the *CDSERV* environment variable points to a directory which contains a file *hepdb.names*, which gives the location of journal files for the various calibration (or other) databases, the locations of these databases, the names of remote servers, and so on.

Users of Objectivity/DB may work with a copy of the boot-file, e.g. if they are at an outside institute - this copy is kept in sync with other copies by the database system. Again, similar functionality is to be found in both FATMEN and HEPDB. In Objectivity/DB, such a "remote institute" is referred to as a *partition* - it has its own consistent copy of the database catalogue and boot-file, its own lockserver, and its own set of database servers.

In contrast to existing systems in HEP, however, a user need not manipulate or concern him/herself with files - the files used by Objectivity/DB to store persistent objects are managed automatically and transparently by the system.

6.6 Objectivity/DB Training

Training on the C++ interface to Objectivity/DB has been offered through the CERN training programme for some time. As this course is best suited to those people who will develop persistent C++ applications, and is less oriented towards end-users, we will supplement this training with an end-user course, based upon the lectures being developed for the 1998 CERN School of Computing.

6.7 ODBMS-based Applications

As described above, numerous experiments at CERN and outside have begun to build applications using Objectivity/DB. The RD45 project has assisted these efforts via informal consultancy (user support) and by providing a centralised installation of Objectivity/DB, information on installation and testing and documentation. New developers are

[8] See, for example, the BaBar introductory guide, available via http://www.slac.stanford.edu/BF/doc/Computing/Databases/www/frames.htm, and the Objectivity/DB technical overview, available via http://www.objectivity.com/Products/TechOv.html.

recommended to follow the Objectivity/DB C++ Developers' Course, offered as part of the CERN training programme. A calibration database prototype has been developed within the CMS collaboration. A similar package has also been developed within the BaBar collaboration. During the coming months, we will evaluate both of these tools with the aim of offering an experiment-independent tool with HepODBMS.

6.8 Conclusions

The recommendations of the LCB referees of the RD45 project have been addressed and the related work described above. In several areas, such as the issue of the ODBMS - MSS interface, there are follow-on activities for the coming year. These will be addressed primarily in conjunction with the proposed production services based upon Objectivity/DB and HPSS.

7 Milestone 1

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

"Demonstrate that an ODBMS can satisfy the requirements of typical simulation, reconstruction and analysis scenarios with data volumes of up to 1TB."

7.1 Requirements

The general requirements for this milestone are based upon those stated in the ATLAS [27] and CMS [28] Computing Technical Proposals, and on discussions at RD45 workshops.

The main requirement for reconstruction is that the ODBMS should be able to keep up with the rate at which data is acquired: 100MB/second for ATLAS and CMS and 1.5GB/second for ALICE. It was agreed that it was not necessary to demonstrate these data rates now, but rather show how such rates could be supported in the future, assuming appropriate hardware resources. Furthermore, it was agreed that an acceptable fall-back solution for ALICE would be to reconstruct in "play-back" mode. In other words, the period when data was not being taken would be used to perform the reconstruction. This results in an effective data rate for reconstruction similar to that required for ATLAS and CMS.

All experiments, as now, plan to perform production reconstruction using a farm. The estimated number of nodes required various from around 100 to 500. It has already been demonstrated in NA45 that up to 32 streams can write into a single Objectivity/DB federation using a lock-free strategy. An extrapolation to 100 streams by 2005, if not much before, seems not unreasonable and hence requires a data rate of just 1MB/second per stream. Even in the case of ALICE, a data rate of 15MB/second per stream would be required, assuming a farm of 100 nodes. Even if the rawdata is entirely rewritten, e.g. for reclustering purposes, rather than added to, the data rates appear rather modest. Hence, the I/O rates for reconstruction purposes are not considered to be a problem.

For analysis, it is assumed that some 150 users will be performing analysis concurrently at any one time per experiment. The data volumes that will be read per analysis are harder to estimate. However, we should clearly exploit the ability of reading just the needed data, and hence minimise I/O. It is also assumed that the data will be distributed over multiple servers - perhaps 100 servers each with 1TB of disk cache and a non-negligible amount of memory. It will be important to exploit the natural parallelism of the database and both disk and memory caching. Simulations of multi-user analysis loads are described in more detail below.

Simulation is not believed to have any special requirements that are not automatically provided for by meeting the requirements of reconstruction and analysis.

7.2 Scalability

The scalability of Objectivity/DB's architecture was covered in detail in the March 1997 status report [4]. At that time, it was not possible to create individual databases (DB) - of which up to 2^{16} are permitted in a federation - larger than 2GB. The current release permits databases up to the limit imposed by the underlying filesystem - essentially unlimited on 64-bit filesystems. We have verified that the 2GB limit no longer exists, by creating databases up to 25GB in size. We have also verified that it is possible to generate federations up to the current limit of 2^{16} databases. The largest test federation that has been created to date has been of the order of 0.5TB - limited by the available diskspace. Plans to build a disk-resident federation of 1TB have been shelved, as the required resources (disks, servers) could not be purchased out of the funds made available. Attempts to build very large federations using HPSS-managed storage are currently under way, and are described further in section 9 on page 62.

Although we have not built a federation larger than 0.5TB, many federations containing over 1000 DBs have been built. Using 25GB DBs, just 40 are required to build a federation of 1TB. Just as, in today's environment, the number of entries in a FATMEN catalogue is entirely independent of the size of the files that are catalogued, the number of databases in an Objectivity/DB federation is totally decoupled from the size of these databases. Thus, the number of databases per federation demonstrated in the various production federations that were built during the last year, such as those generated in CMS test-beam activities (see section 9.4 below), show that federations as large as some 25TB are easily achievable. In practice, building federations much larger than a few hundred GB requires an interface to a mass storage system, as discussed in section 6.2 above. Performance and functionality tests of the current prototype are discussed in section 9.1 below.

We are therefore confident that the requirement of providing federated databases up to 1TB for reconstruction, simulation and analysis can be met.

7.3 Simulation

Neither event nor detector simulation programs generate high I/O rates, nor are these multi-user applications (although multiple analysis jobs may be running in parallel). Hence, the requirements from the point of view of simulation are rather straightforward - the ODBMS must simply support the persistent object model of the application. Although an effort to rewrite the Lund family of event generators has been recently approved, the only simulation program that currently uses an Object Database is GEANT-4. Experience in adding persistency to GEANT-4 was reported on at the last LCB review of RD45. Since that time, further progress has been made, as described below.

The strategy for persistency in the GEANT-4 project splits the persistent data model into two parts. The run based persistent objects, which have to be stored for each run, and the event based persistent objects kept for each event.
The run based information includes run conditions, the geometry definition and the physics process conditions during a simulation run.

The event based part stores information about the primary physics process, a persistent representation of particle trajectories, simulated hits and the resulting digitisation data. An overview of the event based object model is show in the figure below.

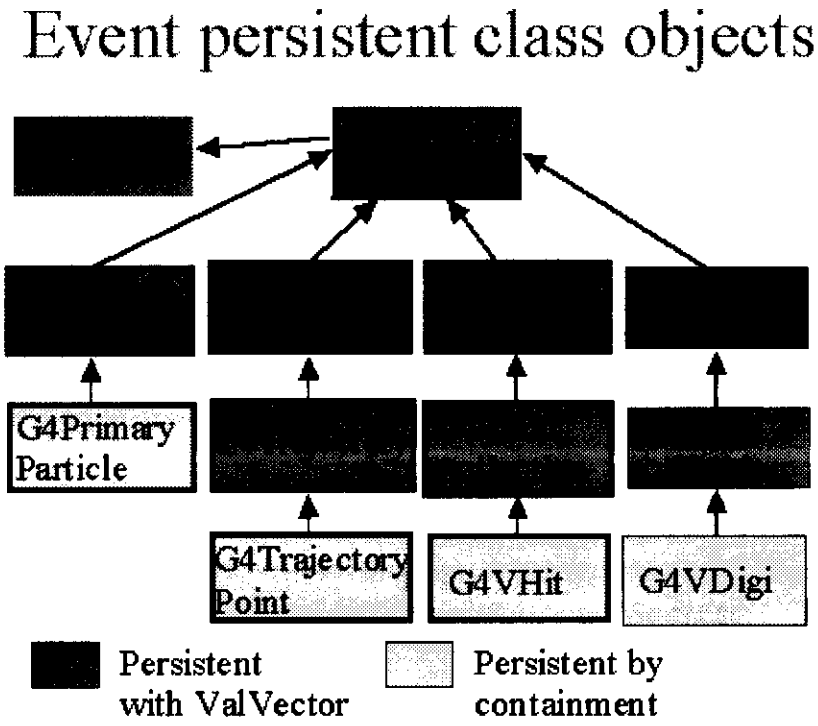

Figure 7 - GEANT-4 Persistent Classes

It should be noted that storage size of classes like trajectory points or single digitisations and hits is rather small in comparison to the overhead for making an object persistent on its own. In the current object model one has therefore chosen to make instances of this relatively small classes persistent by containment in a persistent container. For example the trajectory points themselves are transient objects stored in a persistent trajectory object. Since the object model in the area of event based persistency is relatively simple and the performance penalty introduced by direct persistency has been shown to be negligible, the simulation code will directly make use of persistent objects.

In the area of run based persistency, the GEANT-4 object model is more complex and makes use of many cross-references between the involved objects. The naive approach of simply making some of the classes in the transient model persistent would lead to problems with respect to object lifetimes. In some cases persistent objects would contain references to transient objects that are invalid when retrieved later by another process. In other cases the persistent objects would contain (and therefore store) attributes which are only valid during the actual simulation processing.
On the GEANT-4 mini-workshop on persistency issues held in January 98, it was therefore agreed that a strategy using a parallel hierarchy of persistent objects storing only the

relevant part of the attributes of the original transient model is more appropriate. In this approach a persistent store manager object, which can be extended by the GEANT-4 user, is responsible for mapping a complete tree of transient objects onto an appropriate persistent object tree.
The weaker binding between the transient objects used by the core simulation code and their persistent counterparts will help to decouple the transient and persistent object model models and will reduce the dependency of the core simulation code on the persistency mechanism.
The main drawback of this strategy is the need for a manual synchronisation of both models. If a new attribute is introduced in the transient object model, which also needs to be stored, then the persistent data model has to be extended as well and the mapping code which converts both models into each other has to be updated.

In summary, it is the users of GEANT-4 who will actually design the persistent-capable object model. GEANT-4 itself will provide a skeleton and guidelines based on those of RD45, of how to transfer data from the GEANT-4 transient objects to persistent objects.

7.4 Reconstruction

Production demonstrations of using an ODBMS to provide object persistency for event reconstruction have been made since 1996, by the NA45 experiment. These demonstrations have shown not only the feasibility of using an ODBMS for this purpose, but also some of the advantages of such a system, such as the ability to support multiple concurrent writers, whilst maintaining full consistency. Demonstrations have also been made by CMS, as part of their H2 and X5 test-beam activities in 1997. This work will be continued in 1998, when both ATLAS and CMS plan further test-beam activities based on Objectivity/DB.

Other important activities planned for 1998 include tests by NA45 and COMPASS, in preparation for their high-volume runs in 1999, and the "mock data challenge II", planned by BaBar to run during the second half of 1998.

These activities are described in more detail below.

7.5 Analysis

The issue of data analysis is very closely related to milestone 2, where we address all end-user issues (see section 8 on page 47.) In this section, we address the requirements given in the ATLAS [27] and CMS [28] Computing Technical Proposals. The main requirements stated in these proposals are:

- Support for large numbers of simultaneous users,
- Support for the required data volumes and rates,
- Support for distributed access.

These general requirements are somewhat harder to quantify, as they all depend heavily on the computing model adopted. We therefore make a number of assumptions, based upon current thinking.

It is assumed that some 150 physicists will be actively performing analysis at any time of day or night. Here, "active" is taken to mean the number of users who are reading or writing data to the database within a given time interval - say one hour. Given Objectivity/DB's federated database architecture, it is clearly possible to create a situation whereby many more concurrent users are supported. Theoretically, a total of 2^{16} databases could be stored on separate servers, handling just one user. As the locking granularity is at the level of a container, each single database could support 2^{16} parallel writers without any lock conflicts! In reality, many of the users are likely to be accessing the same data - the "hot" events. Clearly, an important issue will be designing a computing environment such that the individual database servers can provide sufficient bandwidth and whereby the data is efficiently clustered. Current thinking at CERN suggests that data servers will serve a few hundred GB of disk, so as not to limit the I/O throughput. The first such data servers will be installed during 1998. Plans at BaBar are for somewhat more powerful servers, perhaps supporting 1-2TB of diskspace. Nevertheless, the filesystems on these servers should be capable of sustaining data rates in excess of 10MB/second.

Assuming that the primary mechanism by which events are selected is via an event "tag", as described in section 8.4 on page 51, and that an event tag is of the order of 100 bytes, 100GB is then sufficient to store 10^9 event tags. Clearly, this will not be a problem in terms of overall data storage, but both efficient clustering and caching will be needed, to support large numbers of concurrent users. A further option is that of data replication, which could be used both in the local and wide area. Rather than access a remote collection, a user could access a replicated tag database. This would not only reduce the load on any central servers, but would also minimise network traffic. As, in general, we assume that the event data would not be replicated, it would only be necessary to access remote data for the small fraction of events that passed the initial cut on the event tags. Clearly, for certain rare and interesting channels, it would be possible to replicate the full event data too, further reducing the overhead on the wide-area network. Other options, which clearly need to be studied further, include the possibility of using mobile "agents" to perform the analysis close to the data: a classic example of moving the query to the data and not vice-versa. Preliminary studies of using agents in combination with Objectivity/DB have been made, and this area is clearly worthy of further study.

7.6 Simulation of Multi-User Analysis

Tests of the ability of Objectivity/DB to support multiple concurrent users were made using the 256-processor HP Exemplar system at Caltech. This system is built out of 16 nodes, each of which are a 16-processor SMP. Each node is connected to a striped disk array, capable of delivering some 22MB/second. In the tests, the data was stored on two of the nodes (10GB each). The I/O-intensive clients were run on the nodes where the data resided and the CPU-intensive clients were run on the other nodes.

The load was comprised of the following:

- 1/3 of the clients read 10KB objects and computed for 0.001 seconds per object,
- 1/3 of the clients read 100KB objects and computed for 0.1 seconds per object,
- 1/3 of the clients read 500KB objects and computed for 10 seconds per object.

Using the above mix, it was possible to run more than 100 concurrent clients on the system with no degradation in I/O performance. The combined throughput of all clients was essentially constant, at around 18MB/second, up to 100 concurrent clients. These initial results suggest that scaling to 150 concurrent analyses - even without resorting to multiple database partitions and replicas - is achievable today. Support for many more active users will clearly be possible by the time of the LHC production phase.

7.7 Data Reclustering

When using an ODBMS based on a page-server[9] architecture, as is the case with Objectivity/DB, efficient data clustering is important to maximise data rates. In other words, it is important that each data page that is transferred to the client contains a high percentage of objects of interest. The same is true on a coarser scale, when data are cached from tape to disk, by the mass storage system. Not only is it inefficient use of the available disk space to cache unneeded data, but it also results in wasted bandwidth when transferring the data from tape.

Although it is clearly important to cluster data efficiently when it is first stored - both at the page level and also at the level at which data are transferred to/from tape - there will be times when re-clustering is required, to re-establish efficient data access.

The issue of data re-clustering has been studied in both ATLAS and CMS. To study the potential performance gains of re-clustering, a prototype has been developed in CMS. This prototype is based on a mechanism for clustering data into collections, and accessing these collections with read-ahead optimisation. The read-ahead optimisation allows the clustering of different types of objects to be managed in an independent way, and also makes it possible for the "batch reclustering" operation to conserve the database size while preserving optimal throughput. The objects are retrieved through a fast access engine, which uses a schedule to optimise throughput. The schedule only needs to be computed once for every job, and this allows the optimiser to use fairly complex computations.

Subjects for future research are to increase the scalability in the number of access patterns, and an extension of the optimiser to cover migration of collections between tape and disk store.

[9] This is of course also true for other architectures, as the disk I/O is ultimately performed in pages, not single objects.

7.8 Data Import/Export

Objectivity/DB provides a number of facilities for data import/export. For example, a database may be moved from one server to another, within the same federation. This may be performed over the network, if sufficient network bandwidth is available, or offline, e.g. via tape. Applications where this may be useful are for the centralisation of the output of simulation runs. As described above (see section 6.1 on page 21), Objectivity/DB also permits data to be replicated, which could be applicable to calibration data, event tags and so on. However, there may be occasions when the use of a single federation may not be appropriate or the network bandwidth inadequate.

For example, it is likely that developers will wish to work with a private federation, containing a subset of the data in the production federation. This could be required to insulate the production changes from the development environment, or to enable the developer to work disconnected from the network.

In Objectivity/DB, it is possible to copy a database and then attach it to another federation. It is required that the target federation be compatible, i.e. have consistent schema for at least the subset of objects in the databases that are copied, and share database parameters such as the database page size. A copied database may be (re-)attached with a new database ID, in which case the object identifiers of all contained objects are automatically updated.

However, should it be necessary to copy objects which have associations to objects stored in different databases, a more complex strategy is required. The external associations can be simply dropped. This could be appropriate if, for example, the analysis objects are being copied. In this case, it would be possible for any associations to say the raw data to be dropped and tested for by analysis applications.

A more complete solution, however, is to provide a "deep copy" utility, which copies objects and any objects that are referenced. Such a tool has been developed by BaBar to assist in their data import/export, under the assumption that existing networks do not offer sufficient bandwidth to support wide-area data replication (at least not for raw data!). An associated problem is that of maintaining consistency between federations. Here, BaBar intends to use a simple database-ID allocation scheme, which ensures that the database Ids used by different federations are compatible.

7.9 Production Database Services

In today's environment, most of the prototyping and testing using Objectivity/DB is performed against private federations - there is little coordination of issues such as federation identifier, lock-server and so on. Although developers are likely to continue to work with their own federations, this is clearly not appropriate for end users. As part of the 1997 COCOTIME review of computing resources at CERN, it was agreed to establish a number of production database services during 1998. These services would be based around the following components:

- A dedicated server on which the lockserver for the primary partition[10] would run. This machine would also contain the primary copy of the federated database catalog and schema. These servers would have mirrored file-systems for the operating system and database data, in order to offer maximum reliability.
- Data servers, on which the Objectivity/DB server (AMS) would run. These servers would have several hundred GB of diskspace, which would typically be managed by HPSS. At least one data server per experiment would use non-HPSS managed storage, for data that must reside permanently online, such as calibration data, production control data and collections of event tags.

Initially, such services will be established for ALICE (NA45), ATLAS, CMS and COMPASS. These services will help us to gain experience of running Objectivity/DB together with HPSS in a production environment.

7.10 CMS

As described in section 9.4 on page 77, CMS have evaluated the use of an ODBMS at all stages of production including data taking, reconstruction and analysis. Federations of some 1000 databases have been generated at two test beams in 1997, and further work with larger data volumes are planned for 1998.

7.11 NA45

The NA45/CERES experiment is looking for low-mass e^+e^- pairs in heavy-ion collisions at the CERN SPS. At the end of 1995, they took the decision to move to C++ and began to redesign and rewrite their offline software. At the same time, they began to use Objectivity/DB for the storage of some of their data, and are hence the first HEP experiment to have used an ODBMS in production. Although their main production was performed on the CERN CS-2 system, a platform not officially supported by Objectivity, they were able to complete successfully a series of production runs. These runs demonstrated a number of important features of Objectivity/DB, including the ability to support multiple parallel writers. A total of 32 processors were allocated to the NA45 production runs, and each processor was able to write into the database in parallel. This is considered to be an important demonstration of a capability which will be fundamental to supporting the reconstruction farms of the LHC experiments.
Based on the 1995 data, a federated database of approximately 20GB was created, from some 0.5TB rawdata. The 1996 data was similarly processed, resulting in some 50GB stored in Objectivity/DB.

The CERES detector is currently being upgraded to include a TPC, which will improve momentum resolution. A further data taking run is foreseen for 1999, at which stage some 50TB of data will be acquired. It is currently planned that these data will be stored directly in Objectivity/DB, with the data store managed by HPSS. A test run is planned for the end

[10] It is assumed that additional autonomous partitions would be established at a later date, each with its own lockserver and a replica of the federated database catalog and schema.

of 1998, by which stage a reliable version of the interface between Objectivity/DB and HPSS is required.

7.12 BaBar

The BaBar experiment at SLAC will take some 200TB of data per year, starting in 1999. They intend to use a combination of Objectivity/DB and HPSS on which to base their event store.

7.12.1 Introduction

Events are represented as a hierarchy of objects based upon an event header. This header contains references to multiple child objects, each corresponding to a particular processing stage. The information for each processing stage will itself typically be organized as a stage header with references to further child objects. In addition, summary or tag objects are designed to allow for rapid but simple tests to be made on a small subset of the attributes of the children objects without having to access the latter directly. A single event tag is proposed, together with stage tag objects at each of the stages. There is a trade-off between performance and complexity of such queries. Extremely high performance queries can make selections based on the event tag, while less performance but more complexity can result from accessing the stage tags and finally the data from each stage. This strategy can result in dramatic performance gains if appropriate tags are created.

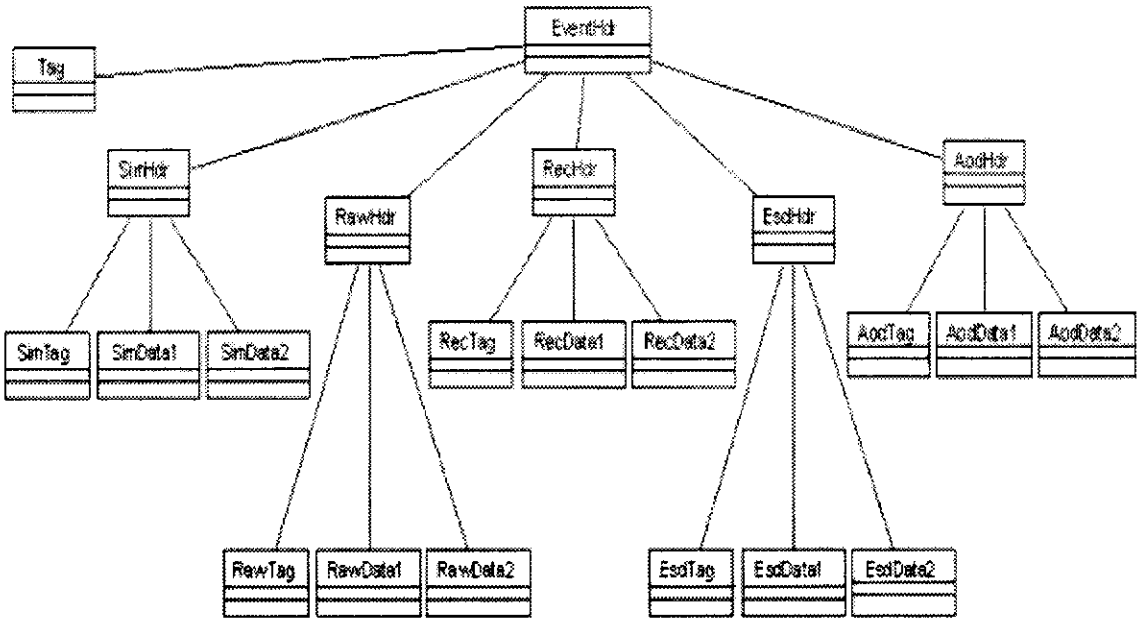

Figure 8 - the BaBar Event Data Model

The data for each event is typically stored across multiple databases, corresponding to the different components. For example, the raw data, reconstructed data, ESD, AOD and

simulated truth would be stored in 5 sets of databases, with the corresponding headers stored in a further 5.

The event header, the stage headers & the various tag objects are located separately from the bulk of the data that makes up the event in order to take advantage of clustering prefetching. In order to allow for separate migration to the mass store, these should be in separate database files rather than separate containers within the same databases. In addition, mapping between groups of file systems (e.g. slow, fast, HPSS) and components gives the possibility of distributing different part of events in a most efficient way depending on the site.

- SIM: Simulated truth (~50KB),
- RAW: Raw data from experiment (~30KB),
- REC: Reconstructed data (~100KB),
- ESD: Event Summary Data (~10KB),
- AOD: Analysis Object Data (~1KB),
- TAG: Event Selection Data (~100B).

7.12.2 Reprocessing

The processing framework consists of an input module that iterates over an input event collection and acts as the head of a chain of processing modules. At the end of this chain, the output module is responsible for the output of the processed information. Conceptually, the output module can be associated with the same or another event collection as the input module.

Processing and reprocessing depends on the prior presence of processed data in the output collection.

1. The output event doesn't yet exist. In this case, a new Event Header (EVT) object is created and added to the output collection. This EVT object references the appropriate Tree Headers of the input event, and has its own copy of the Tree Headers and data nodes corresponding to the generated data.
2. The output event already exists, but does not already contain information for this processing stage. In this case the new information is added as a tree header and associated data nodes under the existing output event header. If the output collection is identical to the input collection, this is equivalent to adding the information to the input event.
3. The output event already exists and already contains information corresponding to this processing stage. In this case we use Objectivity's Object Versioning to create a new version of the appropriate Tree Header and associate the newly generated data nodes with it.

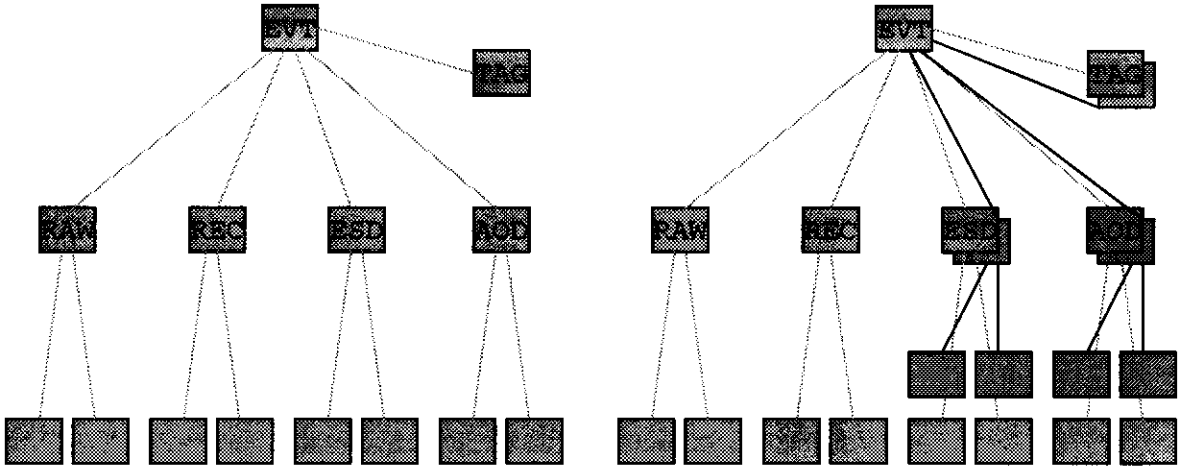

Figure 9 - Reprocessing: same collections

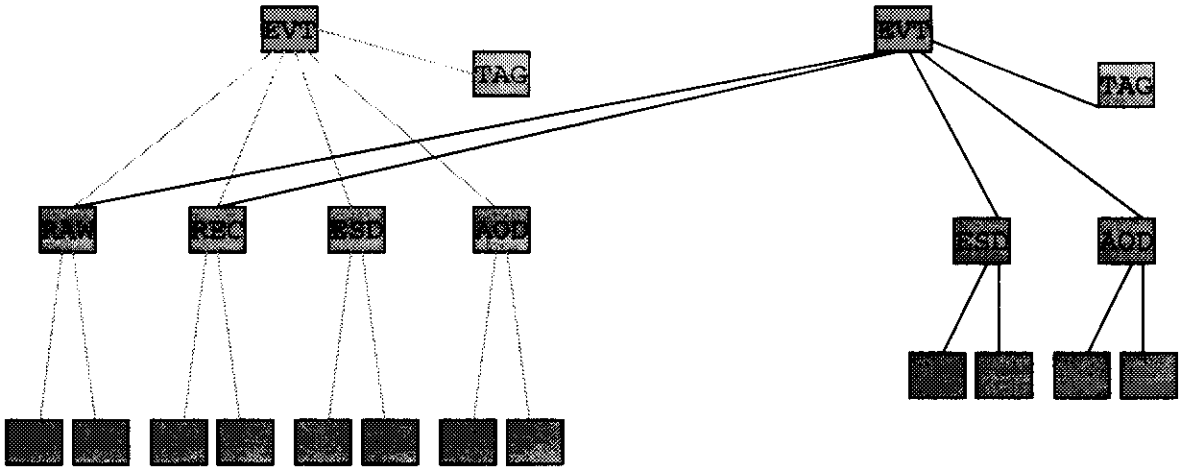

Figure 10 - Reprocessing: different collections

7.12.3 Data Distribution

The BaBar collaboration is establishing a number of regional centres. It is planned that IN2P3 in Lyon will have a complete copy of all of the raw data. Other regional centres, such as RAL and INFN, will have partial copies of the data. Given that some 200TB of rawdata are expected per year, the current network connections between the various sites do not permit distribution of data via this manner. Thus, it is planned that bulk data transfers will be performed by tape, as in the past.

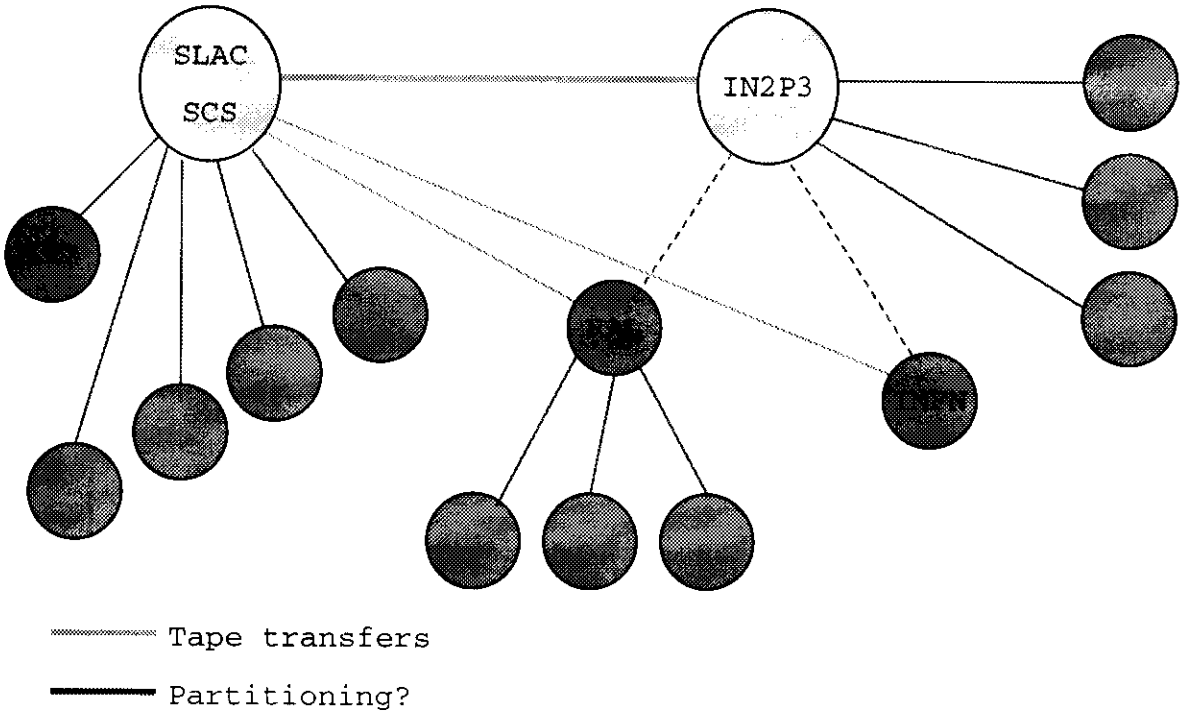

Figure 11 - Data Distribution Strategy

7.13 Conclusions

An ODBMS has been used in production for data taking, reconstruction and preliminary analysis by a number of experiments with federated database sizes ranging from 1-100GB. Additional experience has been gained with the use of Objectivity/DB for providing persistence in GEANT-4, although clearly full production tests cannot be made until at least the open-beta, if not first production version, of GEANT-4 are made available. Both the open-beta and first release are scheduled for 1998. Production federations containing over 1000 databases have been demonstrated, and the scalability of individual databases up to 25GB has been shown. Thus, we believe that an Objectivity/DB federation can today easily scale to at least 25TB, if not well beyond. Scheduled enhancements, including the interface to HPSS, should remove the practical limitations associated with building such large federations. Performance tests, described in section 9 below, have shown that data can be written into Objectivity/DB at close to raw disk speeds per stream. Production tests with NA45 have demonstrated the use of up to 32 parallel streams, making data rates in excess of 100MB/second fully achievable today.

8 Milestone 2

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

"Investigate the impact on the every-day work of the end-user physicist when using an ODBMS for event data storage. The work should address issues related to individual developers' schema and collections for simulation, reconstruction and analysis."

8.1 Introduction

In the current model, the ODBMS that is used for the event data storage is just one component of the overall offline environment. To a large extent, end-users should be unaware of the details of this storage - just as they will be unaware of the storage management software, presumably HPSS, upon which Objectivity/DB will be built. As described in more detail below (see section 6.5 on page 32), access to a given Objectivity/DB federation is determined by the use of an environment variable. By default, a user could even be automatically connected to the production database of the appropriate experiment. Helper classes, distributed as part of the HepODBMS class libraries (see section 10.1 on page 82), reduce the amount of code needed to perform frequently used operations, such as initialising a database session. In the case of interactive analysis, browsers are being developed to permit users to navigate through the database, find an appropriate collection of events, either by name or by characteristics, and use these collections as a starting point for further analysis. Given the power and flexibility of the system, a user will have to know much *less*, e.g. no knowledge of the tape staging system, the run book-keeping system and so forth, but nevertheless be able to work with considerably large data volumes with greater ease.

Users wishing to write or modify persistent applications will require additional knowledge, but again this is much less than in the past. The ODMG language bindings form a natural extension to the language in question - only a few basic principles need to be learnt. This is far from true today, when expert knowledge is required to create and manipulate ZEBRA data stores and data structures.

8.2 Naming

Using an ODBMS the main access method for the end user to retrieve a persistent object is navigation. In other words, the application code directly follows a persistent reference from an event collection object that points to an associated event. Afterwards the user might follow another association from the event object to the raw data for this event. Obviously this approach needs some starting point - in other words, the first persistent object has to be found using another technique than navigation. For example, when a user wants to retrieve a particular collection of events from the database, this could be done by a collection name, which is maintained by the system for all persistent collections. Traditional systems use such a naming technique implicitly through file names for event directories. Since a single

flat naming space tends to be difficult to maintain for a larger set of entities, often the hierarchical name space typically provided by filesystems have been used. In other cases a similar hierarchical name space has been implemented on the application level (e.g. ZEBRA directories for histograms).

ODMG compliant databases typically provide naming facilities on the object level. These facilities allow key objects to be located in the system by a simple name lookup. The implementation of these naming facilities in Objectivity/DB allows using another persistent object, a container, a database or the whole federation as the scope of such a name. The concept of a scope provides a lot of flexibility in object naming. For example, nested and/or overlapping naming schemes are possible. In other words, a single object may be referenced by different names in different scopes. To implement a tree-like structured naming space - like the filenames in a filesystem - is relatively straightforward. A first prototype of such a naming tree is included in the HepODBMS class libraries (see the classes NamedTree and NamedNode). This prototype implements a heterogenous tree of named objects. Users can create "directories" in their private databases and populate them with named objects like histograms or event collections.
The directory structure of the naming tree facilitates the organisation of analysis results from different job runs and simplifies subsequent browsing or visualisation.
Although it is clear that such a simple naming scheme can not solve the problem of how to structure the various meta-data items relevant to analysis, it has the advantage of being familiar to end users.

8.3 Collections

Collections of persistent objects, especially event collections, are of central importance to both the end user and the database administrator.
End users will typically specify their analysis tasks as algorithms applied to a set of input events, such as a named event collection, using a scheme such as that described above. In order not to perform the same selection over and over again, they may also produce a new event collection as result of an analysis job.
Database administrators will try to optimize the overall system performance by redefining the physical clustering of large event collections shared by one or more working groups.
The functional requirements for collections are expected to vary significantly between different tasks. For example, some analysis job will process relatively small collections (less than a few thousand events) but will require maximum speed for the access to individual events. Larger production jobs may need to deal with very large collections (up to $10^9 - 10^{12}$ events) but have less stringent requirements in terms of speed of access.
In most end user cases collections will have to support "overlapping collections". In other words, the same event may "belong" to many collections. These collections will typically be implemented by "reference". In other words, the collection will store persistent references to the events rather than complete copies of events. Very large collections, such as "the collection of all events gathered this year" may have to use a different collection implementation, such as by containment, in order to avoid unnecessary large reference lists. The collection in these cases could be defined as all event objects that are contained in a set of database files.

In order to permit multiple different implementations whilst preserving the same user interface, the use of the "strategy" pattern is proposed [36]. The choice of actual implementation is then transparent to the user, who does not need to know about the internal details in order to use a collection. On the other hand, the creator of new collections will have sufficient flexibility to guide the system by suggesting the most appropriate implementation to be used for a particular application.

The collection interface is designed to closely resemble that of the standard C++ collections or STL (see for example [37].) This will permit algorithms developed for STL containers to also function on event collections. As such, each implementation of an event collection would have to provide at least a basic iterator and a constant iterator, which as a minimum implement the interface of an STL forward iterator.
The "class extent" - the collection of all objects of a given class in the federation - could easily be modelled using a collection by containment.

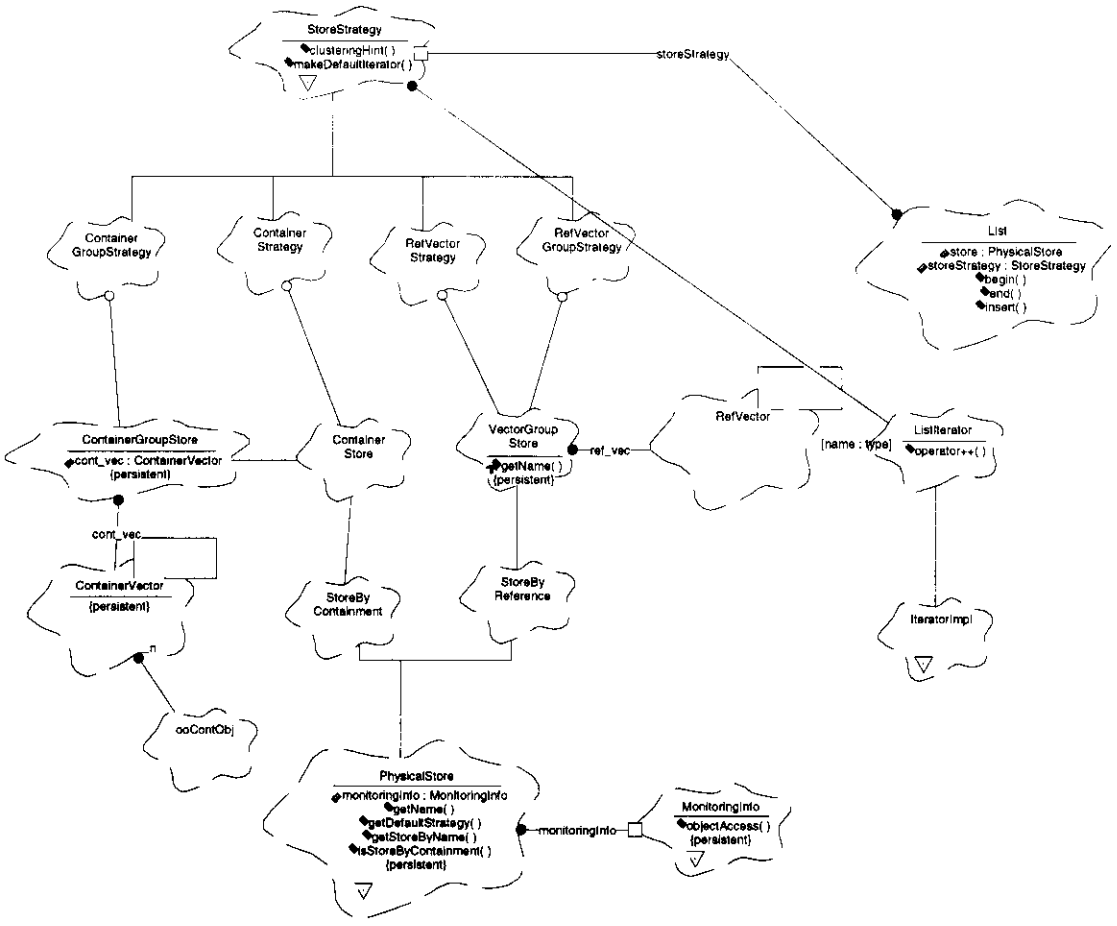

Figure 12 - Class Diagram for Event Collection Prototype

A workshop on event collections was held at CERN during February 1998. The goals of the workshop were to agree on the interface to event collections and on which physical implementations should be made. The target date for an implementation is the 98A release of LHC++, scheduled for June 1998.

The key requirements for event collections identified at the workshop are given below.

8.3.1 Introduction

For the purposes of the following we will use the term event to refer the persistent object class on which the analysis is based. For some experiments this might actually be a pair or track object. The intent is to implement a collection template which allows the concrete class to be supplied as a template parameter, e.g. List<CMSEvent>.

We assume that collections of transient objects, collection of objects within one event as well as transient collections of events are handled adequately by the normal STL and persistent STL implementations. The aim of this implementation is therefore to provide persistent collections of events.

8.3.2 A single class for the user interface

The user should only have to use one single class for both the collection itself and the associated iterator. Different implementations of the physical store will be handled using a strategy object. At collection creation time a hint may optionally be provided, to choose the appropriate implementation with respect to e.g. the expected size.

8.3.3 STL-like interface, including a forward iterator

The collection and the iterator should conform to the collection interface as defined by the STL. At least forward iterators (mutable and const) have to be provided.

8.3.4 Support for collections of up to 10^9 - 10^{11} events

The complete set of events of a given experiment might be very large. For example, the number of events that will be collected by the BaBar experiment will be this order of magnitude. It might not be possible to implement all operations (such as sorting) for very large collections.

8.3.5 Support for a "description" of the collection

It should be possible to decorate the persistent collections with a set of attributes, which further describe it, e.g. a collection name or the selection that was performed to produce the collection.

8.3.6 Set-style operations based on a unique event identifier

Each event provides a unique identifier for which a less operator is defined. This operator will be used, for example, to sort a collection to perform set style operation. To reach acceptable performance we assume that large collections will obey weak set insertion semantics.

The following set operations have to be provided:

- Union
- Intersection
- Difference

8.3.7 Conclusions on Collections

A first prototype of event collection classes, based on the requirements agreed at the February 1998 workshop, will be developed in time for the April 1998 workshop. The goal is to incorporate a version of these classes in the 98A release of LHC++, scheduled for June/July 1998.

8.4 Physics Analysis by End Users

8.4.1 The Traditional Analysis Model

Analysis of physics data is traditionally performed in one of two processing modes loosely labelled as "batch processing" and "interactive analysis". Batch processing is typically used in an initial pass over a large input data set to extract or derive the quantities that will be used for the subsequent physics analysis. This stage may well involve re-running parts of the reconstruction program, possibly with different or improved algorithms or constants. Often the analysis code at this stage selects only a subset of all input events for further analysis. In order to keep the results of such a selection many experiments employ so-called "event directories", which allowed collections of events to be defined and offered facilities for further sub-selection according to a small fixed set of criteria. These criteria are often represented as a bit pattern - the meaning of the bits may even change with time. The main output of batch processing programs was typically one or more Ntuples, which are subsequently used in the "interactive" stage with PAW.

Although both concepts - event directory and Ntuples - offer considerable advantages over what was previously available, a new approach is possible if the data are stored in an ODBMS. For example, if it is realised that an additional variable is required in the Ntuple, the entire Ntuple-generation stage must be rerun. This can be extremely time-consuming, and is somewhat inflexible. Similarly, it is non-trivial to navigate from the data in the Ntuple to the full event data. This can be needed if one wishes to perform a full event display of certain anomalous events.

In summary the main services that have to be provided by an analysis system are:

- Allow the end user to store private analysis quantities in addition to the common attributes stored in the event hierarchy.
- Provide an effective selection mechanism based on private or common analysis attributes.

- Allow visualising private or common analysis attributes.

In addition to allow the end user to conveniently access predefined sets of events and to store private event selection for repeated use, such a system needs to support private and common event collections.

8.4.2 The LHC++ Analysis model

A scheme for performing interactive data analysis has been developed in the context of LHC++. The overall LHC++ strategy, which includes a number of widely-used commercial components, supplemented by HEP-specific extensions, as necessary, offers functionality similar to that of today's CERNLIB. As such, it contains a set of histogram classes, which may be purely transient, written to files or, more conveniently, stored in the same federation as the associated event data. Tools, both batch and interactive, for manipulating these histograms are provided, included mechanisms for fitting the data using Minuit and other minimisation programs. Data can be visualised using industry-standard graphics packages, such as OpenInventor, or manipulated interactively be means of IRIS Explorer, a modular visualisation framework built upon OpenInventor and using components of the NAG numerical libraries.

The following figure shows schematically the difference between the PAW model based on Ntuples and the LHC++ model based on a tagDB. In the latter case, the tag information is still connected to the main event data, and it is possible to navigate from one side to the other.

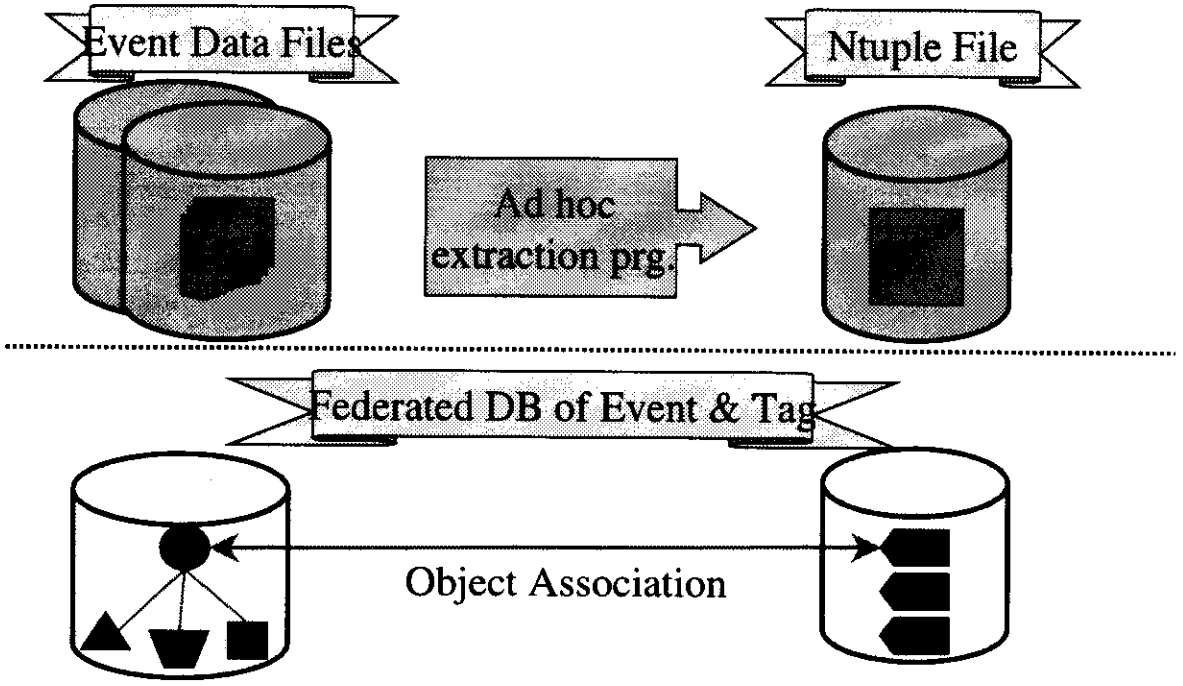

Figure 13 - Tag vs Ntuple Model