

Building a Database for the LHC – the Exabyte Challenge

Jamie Shiers

CERN

1211 Geneva 23

Switzerland

E-mail: Jamie.Shiers@cern.ch

Tel +41 22 767 4928

Fax +41 22 767 8630

Abstract: CERN, the European Laboratory for Particle Physics, is currently building a new accelerator, the Large Hadron Collider (LHC). Scheduled to enter operation in 2005, the experiments at the LHC will generate some 5PB of data per year with data rates ranging from 100MB to 1.5GB per second. Data taking is expected to last 15 or more years, leading to a total data sample of some 100PB. Designing a system that can handle such enormous data volumes implies a solution that can theoretically handle at least one order of magnitude more data than is currently anticipated, namely 1EB (exabyte). Although the production phase of the LHC is still in the distant future, elements of the proposed system have been used in physics experiments at CERN since 1996. In addition, a number of pre-LHC experiments, both at CERN and at other laboratories including SLAC in the US, have adopted the strategy described below. We expect some tens of TB to be stored during 1998 (CERN-NA45) and a few hundred TB per year in 1999 and beyond (the COMPASS experiment at CERN and the BaBar experiment at SLAC).

A strong goal of the project is to use standard, commodity solutions wherever possible. We describe the progress on the project to date, the standards and solutions that are currently being used, performance and scalability measurements as well as plans for the future.

Further information on this project may be found via <http://www.cern.ch/> via the link Research and Development and then RD45.

1 Introduction

In this paper we describe on-going work that is aimed at finding a solution to the data management problems that future experiments at CERN and elsewhere face. The main challenges are the volumes of data (100PB or more) and the timescales of the project (some 25 years in total). Our current activities focus on the use of a standards-conforming Object Database Management System coupled to a Mass Storage System. We show how these two systems can work together offering a solution that is significantly more powerful than a more traditional file-based approach.

2 The Large Hadron Collider (LHC)

High Energy Physics (HEP) can be loosely defined as the study of the fundamental particles that make up matter, and of the forces that act between them. Over the years, the so-called "standard model" has emerged, which comprises 3 families of quarks (up, down, strange, charm, top, bottom) and 3 families of leptons (the electron and its neutrino, the muon and tau ditto). In addition, there are a number of force-carrying particles, such as the photon, Z^0 , W_{\pm} , gluon and graviton. However, a number of mysteries remain, including the origin of mass itself, and the reasons behind the existence of the 3 observed families. Do these hide yet further sub-structure? It is to answer questions such as these that CERN is building the LHC - a proton-proton collider that will be housed in the tunnel currently

occupied by the Large Electron Positron collider (LEP) at CERN, and will enter operation around 2005. The LEP ring has a circumference of 27km and lies some 100m below the surface, straddling the border between France and Switzerland, just outside Geneva. The LHC will host several experiments, including two general purpose facilities, ATLAS and CMS, and an experiment to study the nature of the bottom quark, named LHC-B. The LHC will also be able to accelerate heavy ions, and not just protons, and the ALICE experiment will study the interaction of these ions. The data volumes that will be generated at the LHC will be dominated by the two general-purpose experiments and ALICE - each of these experiments will generate around 1PB of data per year. Including the data from LHC-B, as well as processed and simulated data, a total data volume of around 5PB/year is expected. The LHC itself is expected to run for 15-20 years, giving rise to a total data volume of between 75-100PB. Despite these huge figures, it is unclear which is the more difficult problem - the sheer volume of data involved or rather the long timescales. It is impossible to predict the changes that will occur between now and the end of LHC data acquisition; it is even harder to foresee those that will take place between now and when the LHC starts up, in some 8 years time!

3 The RD45 Project at CERN

In order to investigate possible solutions to these problems, the RD45 project was established at CERN in early 1995. The fact that such a project was established 10 years prior to the first data taking indicates how seriously the problem was viewed within our community. Unlike previous data management projects at CERN, which were largely based on the use of homegrown solutions, often implemented in Fortran, RD45 from the start assumed that the LHC experiments would adopt object-oriented solutions and C++ as the implementation language. This represented a very significant change over previous practice, and opened the door to certain solutions that would not have been possible had we decided to stick with more traditional approaches. Given the very long timescales involved, a first step was the identification of the relevant standards. As indicated by the choice of object-oriented solutions, we were particularly interested in standards related to object persistency - i.e. the ability of an object to persist longer than the lifetime of the creating process. Possible solutions included the Object Management Group's (OMG) Persistent Object Service (POS) and the Object Database Management Group's standards for Object Databases. In this case, the choice was rather straight-forward - the OMG POS was felt by many to be flawed - even unimplementable (it has since been withdrawn!), whereas there were numerous products on the market that claimed to be ODMG-compliant. However, we did not limit ourselves to a study of object databases (ODBMS) as potential solutions. Using books such as Cattell's Object Data Management and Barry's DBMS Needs Assessment as guides, we investigated a wide range of possible solutions, including language extensions, so-called *light-weight* object managers, and full-blown ODBMSs. Based on a rather simple set of initial requirements, such as the need of support for platform heterogeneity and the full C++ object model, we were rapidly able to eliminate both language extensions and object managers. However, there was considerable skepticism within our community that a commercial ODBMS could handle our requirements, if only in terms of data volume. At this point, it is perhaps interesting to note the initial milestones given to the project:

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

1. *A requirements specification for the management of persistent objects typical of HEP data together with criteria for evaluating potential implementations.*
2. *An evaluation of the suitability of ODMG's Object Definition Language for specifying an object model describing HEP event data.*

3. *Starting from such a model, the development of a prototype using commercial ODBMSs that conform to the ODMG standard. The functionality and performance of the ODBMSs should be evaluated.*

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

Milestone 1 was subsequently amended to:

- *Produce statement of the probable capabilities of a HEP persistent object manager based upon commercial object database management systems (ODBMS) and large-market mass storage systems (MSS).*

3.1 Design Principles

A guiding principle behind our activities was the following statement from the ODMG book "The Object Database Standard" :

"The programming language-specific bindings [...] are based on one basic principle: The programmer feels that there is one language, not two separate languages with arbitrary boundaries between them."

This principle contrasts sharply with what was possible previously in HEP. Programs were typically written in Fortran, where as the I/O system was handled by a separate package with a non-intuitive interface. A similar situation is also true when relational databases are used to provide persistency - the programs are written in, *e.g.*, C, whereas the interface to the database is either provided via a proprietary interface, or by embedded SQL, with associated data copies. A general rule of thumb is that approximately 30% of the code is dedicated to this interface, which indeed matches our experience. Aside from the extra development and maintenance involved, an associated problem is the possible lack of data integrity that this involves.

3.2 Using an Object Database

On paper, an ODBMS offers many of the features that we require. The various language bindings that are provided are well integrated with the corresponding programming languages themselves, they offer an extremely rich set of functionality, including support for hardware, operating system and even compiler heterogeneity, schema evolution, object versioning and user data replication. They even offer the promise of language independence - in other words, data that is written today using, *e.g.*, C++, will still be accessible in the future, perhaps using Java. Such features make them extremely attractive as possible building blocks for a multi-PB object store. However, can such systems scale even to the TB range, let alone 100PB? Although there are many examples of production use of object databases, the amount of data stored seems to be small - typically not exceeding a few GB or tens of GB. How can such a solution be applied to a problem many orders of magnitude greater? To explain how we believe that this can be achieved, it is important to emphasize some of the particular characteristics of our data. Firstly, although the volume of data is very large, there are very few dependencies between different parts of the data. To a very large degree, the information recorded from the final-state products of an interaction between two accelerated particles in one of the LHC detectors, each interaction being called *an event*, is completely independent. Furthermore, the data itself is largely read-only and the transaction and concurrency rates extremely low. In addition, there is a very natural

hierarchy within the data. Much of the data for a given event - the raw data - is processed typically once, or once per year, in a sequential manner. Increasingly small subsets of a given event are accessed ever more frequently, depending on the key characteristics of the event. These characteristics in turn provide another way of "sorting" the data - different event categories having markedly varying access patterns and frequencies. Basically, our strategy is to use a single, logical ODBMS that is constructed out of many physical databases - in other words, a distributed object database. The individual databases themselves may be distributed over many database servers in the LAN or WAN. Depending on the characteristics of the data stored in these databases, they may even be offline, e.g. on tape.

3.3 Building a multi-PB Object Store

As described in "Object Databases and Mass Storage Systems: The Prognosis" , we do not expect any major technological problems in building a multi-PB store in 2005. Certainly, in the case of disk storage, the trends are such that very large - perhaps as large as several hundreds of TB - disk farms will be both possible and affordable. Less clear, however, is the evolution of tape technology, as there appears to be little demand for capacity much beyond what is currently available. Despite the trend in disk capacity/\$, we nevertheless feel that a multi-PB disk farm in 2005 can certainly not be assumed. Hence, our strategy is to combine an ODBMS with a suitable mass storage system, and thus permit multi-PB object stores to be constructed. The issues that remain are the selection of the individual components, namely an ODBMS and MSS capable of scaling to the required region, and their integration. This is described in more detail below.

3.4 Tests with Real Data

At the end of 1995, we were contacted by an existing CERN experiment, NA45/CERES, that had just taken the decision to move to C++. They were therefore searching for a solution to their object persistency needs. Although the data volumes that were involved were rather small - a few tens of GB - this provided an excellent opportunity for testing our ideas in practice, with real physics data and a long time before the start up of the LHC. Until then, the tests that had been made were performed using legacy data, which did not have an associated object model, or generated data that did not necessarily correspond to a realistic object model. In addition, we were able to test the impact of providing persistency via an object database on issues such as program design. The results were very largely positive - despite using a version of the operating system (Solaris) that was not supported by the database that we were using, and running on an unsupported platform (Meiko/Quadrics CS-2), we were nevertheless able to use the product in production, and store some 20GB of data. In addition, the data were processed by some 16 nodes running in parallel, something that is not possible on our existing systems, which do not support concurrent write access to individual files.

As a result of these activities, the project was approved for a second year, with the following new milestones:

1. *Identify and analyze the impact of using an ODBMS for event data on the Object Model, the physical organization of the data, coding guidelines and the use of third party class libraries.*
2. *Investigate and report on ways that [ODBMS] features for replication, schema evolution and object versions can be used to solve data management problems typical of the HEP environment.*

3. *Make an evaluation of the effectiveness of an ODBMS and MSS as the query and access method for physics analysis. The evaluation should include performance comparisons with [existing systems used in HEP].*

3.5 Product Choices

As a result of the investigations performed in producing , we were able to identify a number of potential solutions to our problem. Basically, the biggest problem that we needed to solve was that of scale - we needed an ODBMS with architecture capable of scaling to the multi-PB region, and an MSS with similar capabilities. Having ruled out non-ODBMS based solution on functional grounds, we were left with a number of products claiming ODMG compliance. Of these products, we feel that only two have the possibility of scaling to the PB region. The fact that even one such product existed, let alone two, was certainly very encouraging. This is particularly important given the relatively small size of the ODBMS market, and the ODBMS vendors themselves. Indeed, we believe the long-term survival of any given vendor to be one of the largest risks in our strategy. Here, the use of an ODMG-compliant product helps - in theory, at least, an application that uses one of the ODMG bindings can be ported to another compliant product by a simple recompile, and the data itself can be migrated using the ODMG's vendor-neutral exchange format, the Object Interchange Format (OIF). In reality, there are likely to be many issues that significantly complicate such a move. Not least is - again - the volume of data involved. Migrating a few MB or GB of test data is one thing, whereas converting a few PB of data is another. In addition, the ODMG does not attempt to define implementation, but only interface. Hence, the architectures of the various products claiming conformance are very different. Thus, an important aspect of our future work will be to investigate the issues involved in such a migration, which may well be needed given the timescales involved. Some of the issues that need to be addressed include support for data clustering, which will clearly be crucial if one is to achieve efficient access to such large volumes of data.

On the positive side, both of the ODBMSs that we believe to be scalable to the PB region sell strongly into the telecom and financial markets. Whereas their long-term survival can still not be guaranteed, it is unlikely that either of these markets will disappear, and give reason to believe that at least one product meeting the requirements of these markets will continue to exist.

On the MSS side, things are arguably less rosy. Although there are a number of mass storage systems that are available, there is, as yet, no set of standards for the interfaces to such systems, although there is a model - the IEEE reference model on which all recent systems are built. A single MSS, namely HPSS, is designed to scale to the PB region. However, whereas ODBMS vendors count the number of deployed licenses in the hundreds of thousands, this is far from the case with mass storage systems - at least today. The selection of a mass storage system, and the provision of such services at CERN, is outside the scope of the RD45 project. However, an important issue that we do address is the integration of the ODBMS with the MSS. This issue is covered in more detail below.

3.6 Scalability Tests

Having identified a potential solution - at least on paper - it is clear that the overall scalability of the architecture must be verified. At the time of writing, this has only been done using one of the two potential ODBMSs, namely Objectivity/DB. However, by the time of the symposium, we will have repeated these measurements against our alternative solution, namely Versant.

In performing these measurements, we have tested every explicit limit of the architecture of Objectivity/DB, and attempted to find arbitrary, non-documented limits which could constrain us. To explain these tests, a slight digression into the overall architecture of Objectivity/DB is required.

Objectivity/DB supports a so-called *federated database* - in their terminology, this corresponds to a distributed database with consistent, shared schema. In Objectivity/DB, each persistent object has a 64-bit object identifier (OID) associated with it. The 64-bit OID is divided up into 4 fields, each of 16 bits. These sub-fields are used to indicate the (physical) database, the "container" (a set of contiguous pages within a database), the logical page and the slot within the page on which an object resides. Up to 2^{16} databases are permitted per federation, and each database currently maps to a single file. We were able to verify that it was possible to generate 2^{16} databases in a federation, and that individual databases of up to 9GB were possible. This in itself is not a limitation of Objectivity/DB, but is the maximum database size that we believe is reasonable today. The largest federation that we have created to date is 0.5TB - limited by the available disk space. As described below, we have plans for multi-TB test federations and production federations of many hundreds of TB in the near future.

3.7 Enhancement Requests

Although the above measurements confirmed that multi-PB federations are indeed theoretically possible, the maximum achievable size of a federation is clearly limited by practical considerations, such as the maximum database or file size. As a rule of thumb, we estimate that it should be possible to migrate or recall a file in $10^2 - 10^3$ seconds. At 10MB/second, it would take 1000 seconds to recall a 10GB database. Even using techniques such as striping, it is unlikely that one can reduce this time much beyond 10^2 seconds. Using such arguments, we believe that a maximum practical file, and hence database size, is today limited to a few GB. By 2005, it is fairly safe to assume that this will have risen to around 100GB. However, 2^{16} databases of 100GB only permit federations of 6.5PB in size - our requirement is for federations up to 100PB. Furthermore, so as not to introduce any arbitrary constraints, we feel that architecturally, the solution should scale by at least one order of magnitude more than is required, i.e. to 1EB or 10^{18} bytes. Hence, our principal enhancement request concerns architectural changes that would permit EB federations, without, for example, requiring massive files. Essentially, all of the possible options involve permitting more files/federation, by increasing the length of the OID, by allocating more bits to the DBID (fileid) within a 64-bit OID, or changing the logical-physical mapping, such as permitting multi-file databases. We are currently discussing this issue with Objectivity, and hope to have a solution in place by the end of 1998.

3.8 Mass Storage Interface

As described above, we believe that one cannot assume that affordable, manageable disk farms in excess of a few hundred TB will be possible by the year 2005, and hence a means of integrating the ODBMS with a mass storage system is required. The solution that we have identified, and that has been implemented - so far as a proof-of-concept prototype by collaboration between Objectivity and members of the HEP community - is to perform the integration at the level of the Objectivity server. This server is implemented as a page server, and knows nothing about the objects themselves - such knowledge is the responsibility of the Objectivity client. Thus, the server performs basic I/O functions - opening files (databases), seeking to the right offset, reading blocks (database pages) and transferring this information to the client. As such, this offers a fairly natural interface to a mass storage system such as HPSS. Rather than use the standard file system calls, one

"simply" has to replace this layer with calls to the HPSS client API - itself modeled on the Posix filesystem calls. "Heterogeneous" federations are possible, namely ones where some databases are stored in HPSS-managed storage, and others - presumably the bulk of the data - are managed by HPSS. Apart from the need to store some data, such as metadata, permanently online - which could also be achieved using HPSS - it would not be possible, or even practical, to store all of the possible 2^{16} databases in HPSS. One may wish to store some databases locally on systems for which no HPSS client yet exists, some databases will be stored at remote sites, where again one cannot assume HPSS - one can even foresee databases on mobile computers that are not permanently connected to the network.

At the time of writing, the HPSS client is only available for IBM systems, although ports to other platforms are known to be in progress. By performing the interface at the level of the Objectivity server, it is still possible to use non-IBM client machines, including many other Unix platforms and also Windows NT. Furthermore, it isolates the client from the somewhat complex environment that is needed today to support HPSS, including DCE.

3.9 Current Activities

At the time of writing, the focus is on a demonstration that an ODBMS+MSS-based solution can satisfy the key requirements of the entire LHC production chain - from data taking to physics analysis. The current milestones, which need to be addressed by the next project review in April 1998, are given below:

1. *Demonstrate, by the end of 1997, the proof of principle that an ODBMS can satisfy the key requirements of typical production scenarios (e.g. event simulation and reconstruction), for data volumes up to 1 TB. The key requirements will be defined, in conjunction with the LHC experiments, as part of this work.*
2. *Demonstrate the feasibility of using an ODBMS + MSS for Central Data Recording, at data rates sufficient to support ATLAS and CMS test-beam activities during 1997 and NA45 during their 1998 run.*
3. *Investigate and report on the impact of using an ODBMS for event data on end-users, including issues related to private and semi-private schema and collections, in typical scenarios including simulation, (re-)reconstruction and analysis.*

Since these milestones were set, changes to the accelerator schedule at CERN have shifted the above-mentioned NA45 run from 1998 to 1999. NA45 expects to acquire some 30TB of data during this period. In addition, a new experiment at CERN, COMPASS, which is due to start taking data in 1999, has adopted for the same solution, but will have somewhat more demanding requirements. Although 1999 will still be a preliminary run, they expect to take roughly 360TB of data per year when they enter full production, perhaps as early as 2000.

The requirements of the LHC experiments are somewhat harder to quantify. Perhaps the easiest numbers to give are the I/O requirements for data taking and first-pass processing. Here, data rates of 100MB/second need to be supported for ATLAS and CMS, and perhaps as high as 1.5GB/second for ALICE. It is assumed that many systems will be used to perform the processing, with estimates varying from 100-500 nodes. Thus, the data rates per node are somewhat modest - a mere 1MB/second. The ALICE experiment nominally requires much higher data rates, but over a much shorter time period - just one month per year. A viable solution would be for them to process this data at correspondingly slower rates throughout the rest of the year, yielding the same I/O rates - as ATLAS and CMS. Existing experience from NA45 shows that 32 parallel streams are indeed possible, and thus we are confident that we will be able - in 8 years if not before - to handle 3 times as many streams.

Some of the more exotic options that are being considered for the LHC are recalculation rather than storage - the needed data are dynamically and transparently reprocessed using the latest algorithms and calibration constants, and automatic data reclustering based upon access patterns. It is clear that such scenarios could result in very high demands, both in terms of I/O bandwidth and also in available processing power, and have to be studied further. However, more traditional styles of access, with controlled reprocessing and reclustering at perhaps monthly intervals, again requires data rates of just a few MB/stream, but up to 100 or so streams.

3.10 Object Clustering

In Objectivity/DB, complete database pages are transferred between client and server. Not only is the overhead for transferring a complete page rather small compared with the transfer of individual objects, but this also decreases the load on the server systems. In addition, if the page in question contains not only the object that is being requested, but also other objects of interest, there is both decreased server load and network traffic. Conversely, if the page contains just one object of interest, the overhead will increase. Similar arguments are also true concerning the recall of complete databases (files) from tape - if the file being restored contains only a few objects of interest, the overhead will be significant and the system will perform poorly. Thus, it is clearly highly advantageous if efficient data clustering can be performed.

In the ODMG standard, a clustering hint may be given when a new object is created. This tells the database to store the new object "near" another object - be it on the same or adjoining database page, or perhaps just in the same container or database. As this hint is given on a per object basis, it is possible to define complex clustering strategies. For example, it is possible to store the various objects that make up an event and even the objects that make up events with certain characteristics according to different policies. Of course, it is not possible to define a strategy that is ideal for all possible types of access. However, by optimizing for the main access patterns, the overall efficiency of the system can be greatly increased.

3.11 Metadata

Although usually described as being "data about data", we define "metadata" to mean "data that makes other data useful/usable". There are numerous examples of such data in HEP, ranging from calibration data, needed to correct for variations in the response of the detectors, to the schema of the various persistent objects, to data that helps users find the data that they are interested in. Clearly, metadata and the corresponding data are associated with each other. When using an object database, such relations can be represented directly - by associations between the metadata objects and the data themselves. Although there are sound arguments for the separation of data and metadata, these arguments are valid for physical, rather than logical, separation. Clearly, it is not useful to have to restore a 100GB database from tape simply to discover from the metadata that the data are in fact not needed. However, maintaining the logical connection between the two has immediate advantages. Firstly, navigation from the metadata to the data is trivial. Secondly, if implemented using bi-directional associations, then the database itself can ensure consistency. Neither of these is true if separate systems are used.

Calibration data are expected to total some 100GB per year (compared to 1PB of data). The schema themselves will not represent a significant amount of data - it is not yet clear how many persistent-capable classes there will be, but it is likely to be of the order of 10^3 . Metadata are expected to be widely used as the entry point into the system. We expect users and analysis groups to store collections of pointers to objects that correspond to specific

event selections. The criteria used to make these selections will also be stored in the database. We anticipate that these event collections will be named according to some simple - probably hierarchical - naming schema. This will allow straightforward access to collections of events, e.g. "MyCollection" or "Higgs_Events". It will also permit selection based on the criteria used to define these collections. "MyCollection", for example, could refer to all events with a transverse energy greater than a certain quantity and 2 electron candidates. Not only is it useful to have the database manage the definition of "electron candidate", which may well vary according to context, but such a collection is clearly a better starting point for a selection requiring the same E_t cutoff but 4 electron candidates than the entire 100PB store!

3.12 HEP-Specific Extensions

Although an ODBMS offers a rich set of features, we have found a number of areas where extensions have been necessary. Largely speaking, these fall into two categories:

1. Database Administration (DBA) Tools.
2. Class libraries that extend the programming interface.

The DBA tools that have been developed within the RD45 project are written using the Java ODBMS binding. Although the basic functionality is also provided via "command-line" tools and also through the various language bindings, it can be convenient to provide a more user-friendly interface, particularly when dealing with some of the more complicated options, such as data replication. These tools also allow for comprehensive error checking and allow for site customization. For example, data replication, which is today supported in Objectivity/DB at the level of a physical database, is likely to be managed according to a specific policy, which is clearly unknown to the database vendor. Given the large number of databases and sites that will exist, it would be extremely inconvenient to manage replication with the low-level tools that are available. These require the DBA to explicitly state which database(s) should be replicated to which node(s). A more likely scenario is that a set of databases, *e.g.*, those corresponding to a certain physics channel, will be replicated to a set of sites, *i.e.*, the ones participating in the analysis in question. Clearly, it is relatively simple to build a tool with an intuitive drag-and-drop interface that uses the database itself to manage the necessary configuration information.

The class libraries that have been developed so far help to minimize the dependence on a given database vendor and smooth out the changes between various releases. In addition, they reduce the changes that are necessary in the conversion of a transient application to a persistent one. Most importantly, they provide support for sophisticated data clustering strategies, event collections and associated metadata, and so forth. They have been used, for example, in the NA45 production runs to provide a lock-free strategy that permitted up to 32 nodes to write to the database in parallel. In the past, such an operation would have required the use of separate files with an expensive "merge" step at the end.

3.13 Production Plans

Although the production phase of the LHC is still many years away, large-scale production is expected to start in early 1998. In addition to ongoing work with the NA45 and other experiments, the LHC experiments are expected to increasingly use the ODBMS+MSS solution for the storage of simulated data and for test-beam runs. To accommodate these production plans, we intend to set up production services for 4-6 different experiments from the beginning of 1998. Each of these experiments will use a separate Objectivity/DB federation, with its own set of servers. Initially, we will establish a single lock-server per experiment, with one or more data-servers, each handling a few hundred GB of disk space,

as required. At a later stage, the data-servers will also run HPSS disk movers, and will be connected to the tape servers - shared across the different experiments - by HiPPi-class networks. Explicitly, we expect a production version of the Objectivity/DB - HPSS interface no later than Q4 1998.

3.14 Comparisons with Existing Systems

The systems that are used in today's high energy physics experiments provide considerably less functionality than is described above. Not only are they poorly integrated with the programming language, but a variety of separate packages, with inconsistent interfaces, are used for different aspects of the problem. For example, to locate the raw data corresponding to the events that give an anomalous signal in a particular analysis requires an ad-hoc and labor-intensive process. Using an integrated approach based upon an object database, such an operation is trivial.

A good example of the manpower savings made possible by using an object database is that of a calibration management system. Existing experiments have typically implemented their own systems. These systems, which are significantly less complicated than a complete data management system, have been estimated to take up to 10 man-years, of highly experienced personnel, to write and 1-2 full time equivalents to support. Using an ODBMS as a basis, a working system can be built by a student in less than one year. Indeed, it is believed that the effort previously required within an experiment to support just their calibration data may well be sufficient to manage their entire data sample!

4 Summary and Conclusions

We have demonstrated the feasibility of using off-the-shelf, commercial solutions, as the building blocks for a multi-PB distributed object store for HEP data. Although a small amount of code has had to be written to integrate and extend the various components, the savings with respect to "roll-your-own" solutions are considerable. Not only does such a solution offer considerably more functionality than existing, home-grown solutions, but it is also markedly more scalable.

5 Acknowledgements

The work described in this paper is the result of many people's effort over several years both within the RD45 project and outside. Eva Arderiu Ribera wrote the DBA tools described above. Dirk Düllmann largely wrote the class libraries, developed to extend the functionality of the ODBMS, with contributions by Marcin Nowak and others. Some of the classes were based upon examples obtained from Objectivity. Objectivity and Andy Hanuchevsky of SLAC implemented the interface between Objectivity/DB and HPSS. Members of the CERN IT/PDP group run the HPSS system installed at CERN. Finally, we owe much to the NA45 collaboration, who were brave enough to entrust their data to something new.

6 References

- [1] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1998, CERN/LHCC 98-x
- [2] Using an Object Database and Mass Storage System for Physics Production, March 1998, CERN/LHCC 98-x
- [3] RD45 Project Execution Plan, 1997-1998, April 1997, CERN/LCB 97-10
- [4] RD45 - A Persistent Object Manager for HEP, LCB Status Report, March 1997, CERN/LHCC 97-6

- [5] Object Databases and their Impact on Storage-Related Aspects of HEP Computing, the RD45 collaboration
- [6] Object Database Features and HEP Data Management, the RD45 collaboration
- [7] Using and Object Database and Mass Storage System for Physics Analysis, the RD45 collaboration
- [8] RD45 - A Persistent Object Manager for HEP, LCRB Status Report, March 1996, CERN/LHCC 96-15
- [9] Object Databases and Mass Storage Systems: The Prognosis, the RD45 collaboration, CERN/LHCC 96-17
- [10] Object Data Management. R.G.G. Cattell, Addison Wesley, ISBN 0-201-54748-1
- [11] DBMS Needs Assessment for Objects, Barry and Associates (release 3)
- [12] The Object-Oriented Database System Manifesto M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik. In Proceedings of the First International Conference on Deductive and Object-Oriented Databases, pages 223-40, Kyoto, Japan, December 1989. Also appears in 13.
- [13] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1, 1991.
- [14] Object Management Group. Persistent Object Service Specification, Revision 1.0, OMG Document numbers 94-1-1 and 94-10-7.
- [15] The Object Database Standard, ODMG-93, Edited by R.G.G.Cattell, ISBN 1-55860-302-6, Morgan Kaufmann.
- [16] ATLAS Computing Technical Proposal, CERN/LHCC 96-43
- [17] CMS Computing Technical Proposal, CERN/LHCC 96-45

