6 Milestone 3

The third milestone set at the April 1998 review of the RD45 project was as follows:

"Develop and provide production versions of the HepODBMS class libraries, including reference and end-user guides."

6.1 Introduction

The HepODBMS class library provides an insulating layer that completes and extends that defined by the ODMG [22]. On the one hand, it extends vendor compliance and minimises the application developer from minor API changes in different releases of the product, whilst on the other it extends the standard to provide a more complete interface, particularly in areas of concern to the HEP community.

The HepODBMS class library is released and distributed as part the LHC++, ensuring consistency between the library itself, the version of the underlying database (Objectivity/DB) and applications that are built on top of the library.

The main areas of activity concerning the HepODBMS libraries over the review period have been:

- Requirements gathering, design and implementation of scalable event collections.
- Revised implementation of the existing naming scheme.
- Integration of the BaBar "conditions" database application.

The user guide and reference manuals have been updated to reflect these changes. The user guide is marked up in XML, allowing versions optimised both for the screen (currently HTML generated automatically from the XML) and printer (currently PostScript) to be generated from a single source. The reference manual is generated automatically from the source code itself, using the DOC++ scheme. In addition, as is now the standard with all LHC++ components, "CERNLIB-style" short-writeups have been produced for all packages, facilitating keyword-based search and integration into the existing CERNLIB documentation scheme.

6.2 Event Collections

The C++ standard includes the definition of a set of collections based upon the work of Alex Stepanov and Meng Lee. These collections are frequently referred to as the "Standard Template Library" or STL [33], although the more correct term is simply the Standard Library.
The STL includes the following:

- Containers – Generalized classes used to hold a collection of typed data.
- Iterators – Generalized pointers used to traverse the elements of a container.
- Algorithms – Procedures that can be applied on all or part of a container.
- Function Objects – Behaviors used as a parameter by a container or algorithm.
- Allocators – Objects responsible for allocating storage.

The ODMG standard includes STL-compliant collection classes. For the main STL collections, a persistent equivalent exists, designated by a leading `d_`.
Objectivity/DB provides persistent STL collections classes based upon the ObjectSpace implementation - also available in transient form via LHC++.

In addition to the persistent STL collections described above, HepODBMS provides highly scalable collections - capable of handling very large numbers of entries. They were designed to handle up to 1000 million objects, which would not be possible using the standard persistent STL classes offered with Objectivity/DB.
HepODBMS defines following templated collection that is usable for any kind of persistent objects:

```
typedef h_seq<Event> EventCollection;
```

The interface of this collection offers the following:

- STL interface independent of implementation.
- Single User visible collection class : `h_seq<T>`
- Single STL like iterator: `h_seq<T>::iterator`
- Uses hybrid of templated classes and delegation.
- User extensible through strategy objects.

For example, we may wish to access an existing event collection by name. We first define the collection, as follows:

```
EventCollection evtCol("/usr/dirkd/collections/myEvents");
```

We then need to define an *iterator* for this collection (STL-like):

```
EventCollection::const_iterator it;
```

We can now iterate through the collection and read individual event objects:

```
it = evtCol.begin();
while( it !=  evtCol.end() )
    {
 cout << "Event: " << (*it)->getEventNo() << endl;
     ++it;
    }
// support for (some) STL algorithms
```

```
 int cnt=0;
 count(evtCol.begin(),evtCol.end(),1,cnt);
```

Writing to the collection - e.g. adding new events - is shown below.

```
HepRef(Event) evt;
for (int i=0; i<500000; i++)
    {
    // create a new event using the clustering hint of the sequence
    evt = new(evtCol.clustering()) Event;

    // store the new object ref in the sequence (only needed for ref
collections)
    evtCol.push_back(evt);

    // fill the event
    evt->setEventNo(i);
  }
```

The following example shows how the HepODBMS collection class may be used to store events. It is available via
/afs/cern.ch/sw/lhcxx/share/HepODBMS/examples/createCollection/createCollection.cpp

```
#include "HepODBMS/clustering/HepDbApplication.h"

#include "EventSeq.h"

d_Ref<ooObj> VStore<Event>::store_clustering;

void print_evt(d_Ref<Event> evt)
{
  cout << "Event Nr=" << evt->getEventNo() << endl;
}


int main(int argc, char *argv[])
{
  //
  // Persistent event collection tests.
  //
  HepDbApplication app;

  app.init();

  app.startUpdate();

  ooHandle(ooDBObj) db_h = app.db("Sequences");
  ooHandle(ooContObj) ooc_h;

  ooHandle(ooContObj) eventCont = app.container("Events");
  ooc_h= app.container("Stores");

  ooDelete(eventCont);
  ooDelete(ooc_h);

  VStore<Event>::store_clustering = app.container("Stores",1);
  eventCont = app.container("Events");
  app.commit();
```

```
  cout << "created db and container" << endl;

  app.startUpdate();

  h_seq<Event> seq("dirks","vector");

  app.commit();
  cout << "created store object" << endl;
  app.startUpdate();

  HepRef(Event) evt;

  for (int i=0; i<5000; i++)
  {
    // create a new event, use the clustering hint of the sequence
    evt = new(seq.clustering()) Event(i);

    // store the new object ref in the sequence (only needed for ref
collections)
    seq.push_back(evt);

  }

  app.commit();

  return 0;
}
```

6.3 Naming and Meta-Data

The naming facility provided by HepODBMS allows a *name* to be associated to any persistent object that is stored in the database. A name is simply a text-string that is associated with an object. Of course, it is not intended that *every* object in the database is named - this would not be efficient and would have a significant overhead. Naming is most useful for defining "entry points" into the database. For example, a collection of events could be given a name - such as "Higgs candidates". As a flat naming scheme is often inadequate - and would clearly not be useful in a multi-user system, HepODBMS provides a hierarchical naming scheme, similar to that of a Unix filesystem.

The `HepNamingTree` class provides the following Unix-like methods:

- makeDirectory(path),
- changeDirectory(path),
- removeDirectory(path),
- nameObject(objRef,path),
- findObject(path),
- removeName(path),
- removeObject(path),
- startItr(),
- nextItr()

The usage of these methods is shown in the following example. The `HepDBApplication` class automatically places the application in the "directory" corresponding to the current username – for example: /usr/dirkd – to avoid possible conflicts between the naming trees of different users. Naturally, it is possible to navigate to any point in the naming tree.

```
typedef h_seq<Event> EventCol;

// initialize DB session
HepDbApplication app;
app.init("fdBootName"); // implicit cd /usr/$USER/
// move to test-beam
app.naming.changeDirectory("test-beam");
evtCol = EventCol::findByName("inputEvents");
EventCol::iterator it;
for (it = evtCol.begin(); it != evtCol.end(); it++)
{
    -- do something--
}
```

6.4 Conditions Database

The basic features of the BaBar conditions database were described in the RD45 Status Report submitted to the LCB in April 1998 [1]. Essentially, calibration objects are inserted into the database with a specified validity range (valid from – to), and then retrieved by validity instant. Multiple calibrations may exist for a given instant, default calibrations for such an instant may be set, or the user may choose a calibration explicitly.

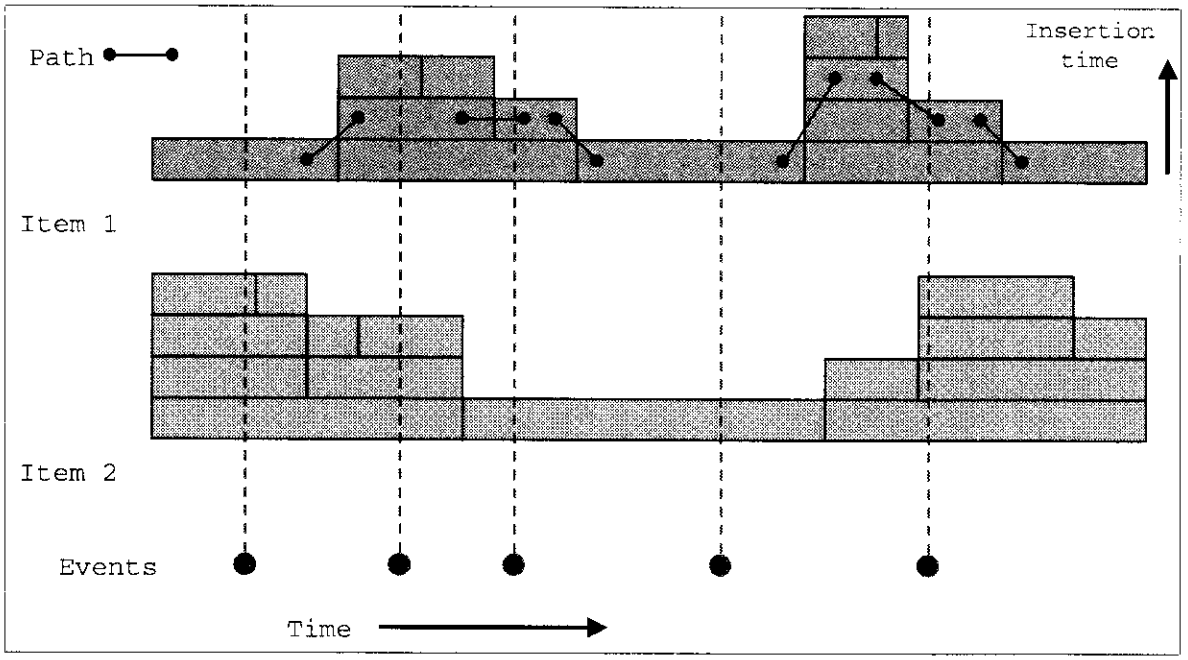

Figure 19 - Multiple Calibration Objects

In order to introduce the conditions DB package into HepODBMS, dependencies on the BaBar environment and Rogue Wave Tools.h++ classes were removed – replaced where appropriate by the corresponding ObjectSpace classes. An example of such a class is *HepTime*, based on the ObjectSpace 64bit time classes (thus avoiding Y2K-like problems until the year 32766AD).

The current version of the conditions database is used by the NA45 experiments and it is being evaluated by ATLAS, CMS and LHCb. The original version of the system is in production use at BaBar, where further enhancements have been made. It is foreseen that these enhancements be integrated into the HepODBMS version. Additional enhancements include the possible provision of a "global tag", covering calibrations from multiple sub-detectors. However, more discussions on the exact requirements are needed before design and implementation.

6.5 Conclusions

Production releases of HepODBMS and associated manuals have been made. The library will continue to be ported to new releases of Objectivity/DB and enhancements will be provided as required. The bulk of the library is expected to be rather stable for the immediate future, although further work on the conditions DB is expected, as described above.

7 Milestone 4

The fourth milestone set at the April 1998 review of the RD45 project was as follows:

"Continue R&D, based on input and use cases from the LHC collaborations to produce results in time for the next versions of the collaborations' Computing Technical Proposals (end 1999)."

An initial list of R&D activities, proposed by the LCB referees, is given below:

- Database usage over a wide area network.
- Clustering and re-clustering strategies.
- Database integration with mass storage systems.
- Multi-user, multi-federation issues.

7.1 Wide-Area Database Usage

We report below on a number of cases where Objectivity/DB has been used in the wide-area.

7.1.1 Reconstruction of Simulated CMS Events

As part of the GIOD project[12], roughly 600,000 fully-simulated multi-jet QCD events where generated on the Caltech Exemplar system using the CMS simulation program, CMSIM. This production resulted in some 600GB of data, stored in ZEBRA FZ format in the Caltech HPSS system. Some 400,000 events where shipped on tape to CERN and reconstructed on 10 PCSF nodes in parallel, using the CMSOO program. The resultant data was stored in an Objectivity/DB federation of around 32GB in total. The individual database files – each some 200MB in size – where transferred back to Caltech using *ftp* and attached to a local federation using *ooattachdb*. The steps involved are shown in Figure 20 below. The data rate achieved for the trans-atlantic ftp was approximately 11GB/day or ~1TB/year. Scaling to a 622Mbps link, approximately 1PB/year should be achievable.

This test did not make direct use of Objectivity/DB support for WAN usage, such as writing remotely to the Caltech federation from the PCSF nodes at CERN, or the *oochangedb* utility to move databases from one node to another. Nevertheless, it demonstrates a pragmatic solution to the use of Objectivity/DB in the wide area, re-using standard tools and minimising risk.

[12] See http://pcbunn.cithep.caltech.edu/Default.htm.

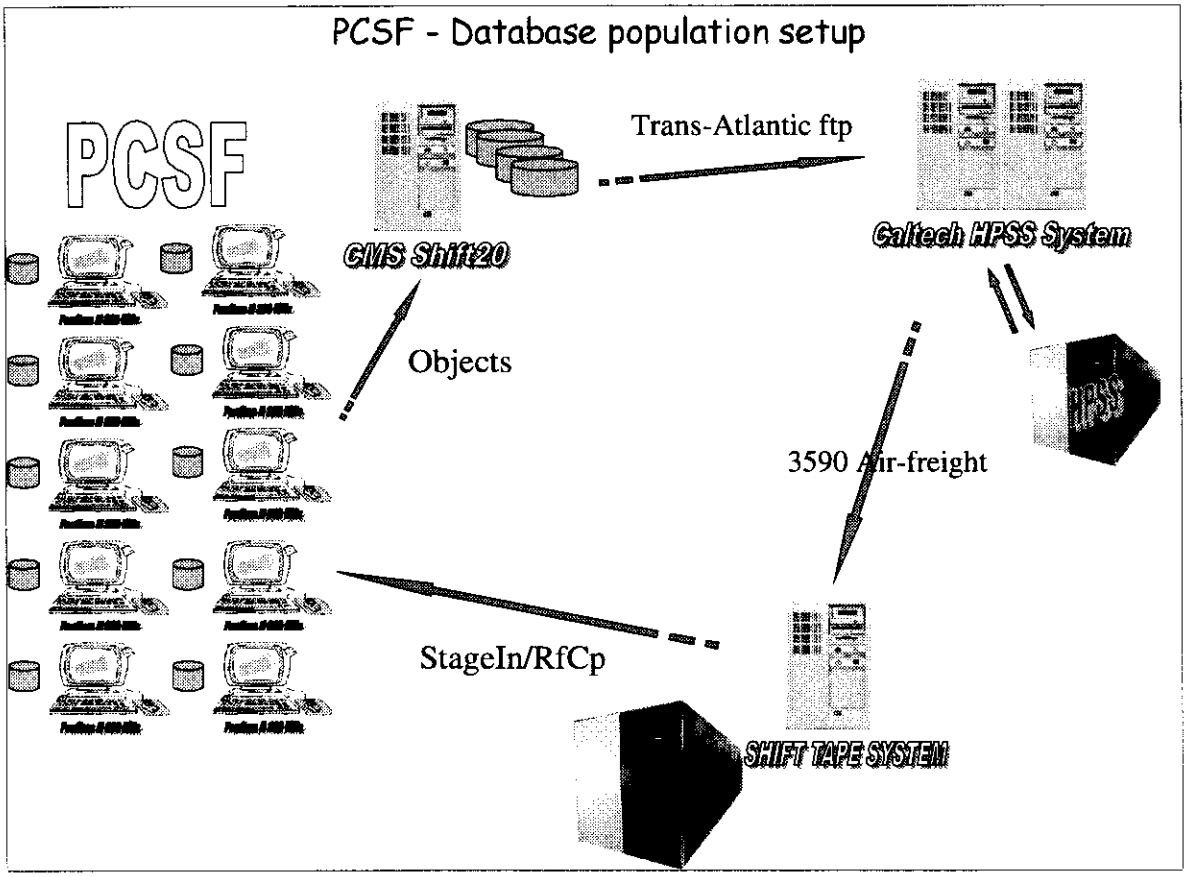

Figure 20 - GIOD Data Flow

7.1.2 WAN Tests Between CERN and KEK

An Objectivity/DB test-bed has been established at KEK, as part of the MONARC test-bed activities. Using this facility, a test of Objectivity/DB Data Replication (DRO) has been made, including performance comparisons between terrestrial and satellite based 2Mbps links[13]. The tests included two scenarios:

- The database is first populated and then replicated, i.e. objects are written locally into the database, and then the entire DB replicated.
- The database is first replicated and then populated, i.e. each update transaction updates both local and remote images.

These tests show that the basic DRO protocol functions correctly in the WAN, but that rather extensive hand-shaking occurs. In addition, the packet size, which appears to be the page size for data transfers, and 4172 bytes for "system" transfers, is not optimal for networks with long round-trip times.

[13] See http://wwwinfo.cern.ch/asd/rd45/workshops/july99/wan-dro/index.htm.

7.1.3 MONARC

In the framework of the MONARC project, a testbed working group has been setup. The goals of the working group include:

- Gathering experience in tuning Objectivity/DB for optimal performance.
- Measuring the response time of the key components (AMS server and networks).
- Validating the assumptions made by the MONARC simulation working group.
- Measuring the effects of different data models and data access patterns.
- Demonstrating the feasibility of physics analysis using an ODBMS distributed in the WAN.

The sites currently participating in the MONARC testbed working group and the associated resources are shown in Table 3 below.

CERN	SUN Enterprise 450 (4*400MHz CPUs, 512MB memory, 4 UltraSCSI channels, 10*18GB disks) Use of mass storage management (HPSS) facility is being planned.
Caltech	HP Exemplar SPP 2000 (256 CPUs, 64 GByte memory) HPSS (600 TB tape + 500 GB disk cache) HP Kayak PC (450 MHz, 128 MB memory, 20 GB disk, ATM) HP C200 (200 MHz CPU, 128 MB memory, 10 GB disk) Sun SparcStation 20 (80 GB disk) Sun Enterprise 250 (dual 450Mhz CPUs, 256 MB memory[1]) Micron Millennia PC (450 MHz CPU, 128 MB memory, 20 GBytes disk) ~1 TB RAID FibreChannel disk (to be attached to the Enterprise 250[1]) [1] shortly to be ordered
CNAF	SUN UltraSparc 5, 18 GB disk
FNAL	ES450 Sun Server (dual CPUs), 100 GB disk + access to a STK Silo
Genova	SUN UltraSparc 5, 18 GB disk
KEK	SUN UltraSparc, 100 GB disk
Milano	SUN UltraSparc 5, 18 GB disk Access to non dedicated facilities is available at CILEA: to a SUN system similar to the dedicated one and to the HP Exemplar SPP 2000 of the Centre, for agreed tests.
Padova	SUN UltraSparc 5, 117 GB disk + SUN Sparc 20, 20 GB disk
Roma	SUN UltraSparc 5, 27 GB disk
Tufts	Pentium II 300 MHz PC, 12 GB disk (+ Pentium-II 400 MHz PC, 22 GB in July)

Table 3 - MONARC Test-bed Systems

7.2 Clustering and Re-clustering Issues

The ODMG standard allows for a "clustering hint" that can be specified when a new instance of a persistent-capable object is created. This allows an application to request that the new object be stored physically close to an existing object – typically on the same or adjacent database page. As disk and network I/O typically takes place in chunks that are larger than the size of individual objects, I/O can be reduced by taking advantage of such clustering. For example, if objects A and B are stored on the same page, a request to access object A will automatically bring object B into the client cache. A subsequent attempt to access object B will be satisfied from memory, thus avoiding an explicit I/O for object B. Support for object clustering is provided directly by Objectivity/DB and further extended via the HepODBMS layer[14].

In the HEP environment, data analysis is often increasingly selective: subsequent analyses often access smaller and smaller subsets of the data, resulting in fragmented access. To avoid such fragmentation, data were typically distilled into smaller and smaller subsets, implying redundant data copies that some became out of date. This problem is shown schematically in Figure 21 below, where different jobs read increasing small subsets of events.

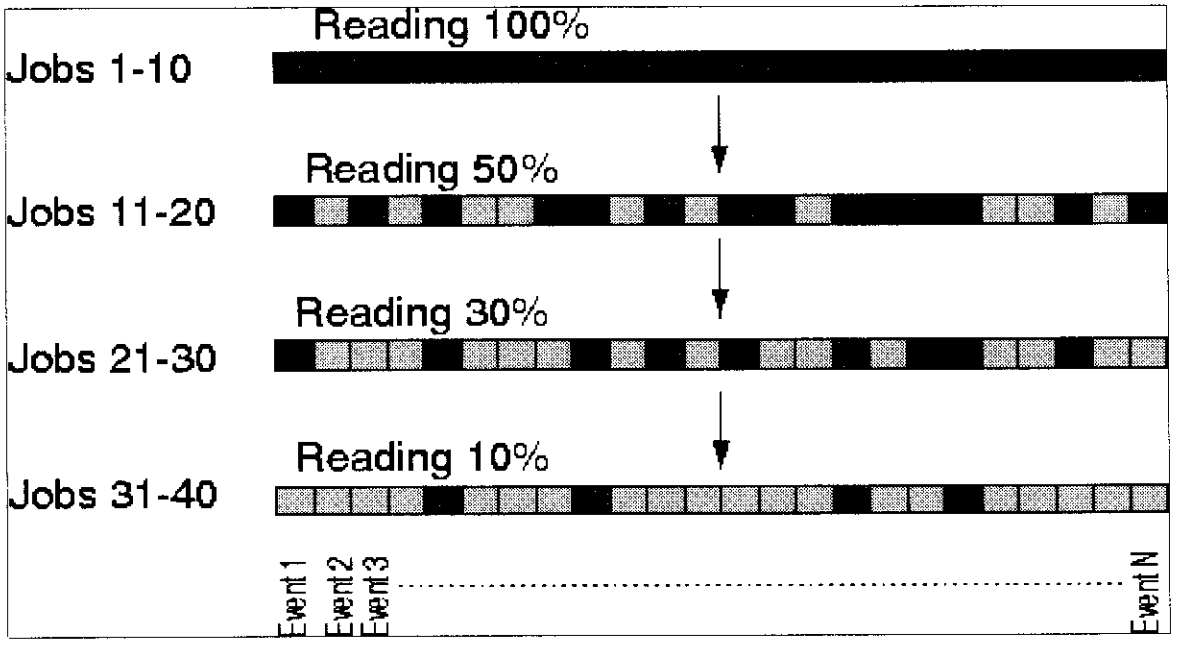

Figure 21 - Data Clustering and Increasing Selectivity

To improve the clustering for jobs that access only a small fraction of the data, one can simply copy or move the accessed events so that they are contiguous, as shown in Figure 22.

[14] See http://wwwinfo.cern.ch/asd/lhc++/HepODBMS/user-guide/ho.html.

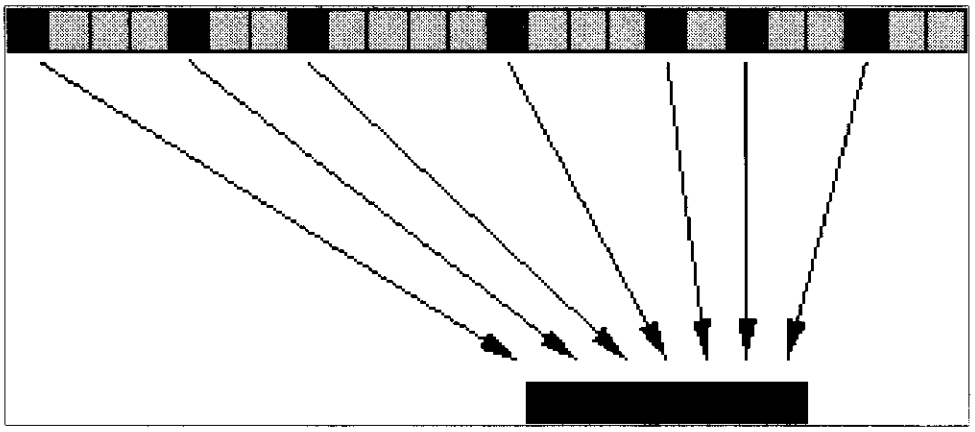

Figure 22 - Reclustering Operation

Such reclustering has been studied in the CMS collaboration [38], where data are dynamically reclustered based on observed access patterns.

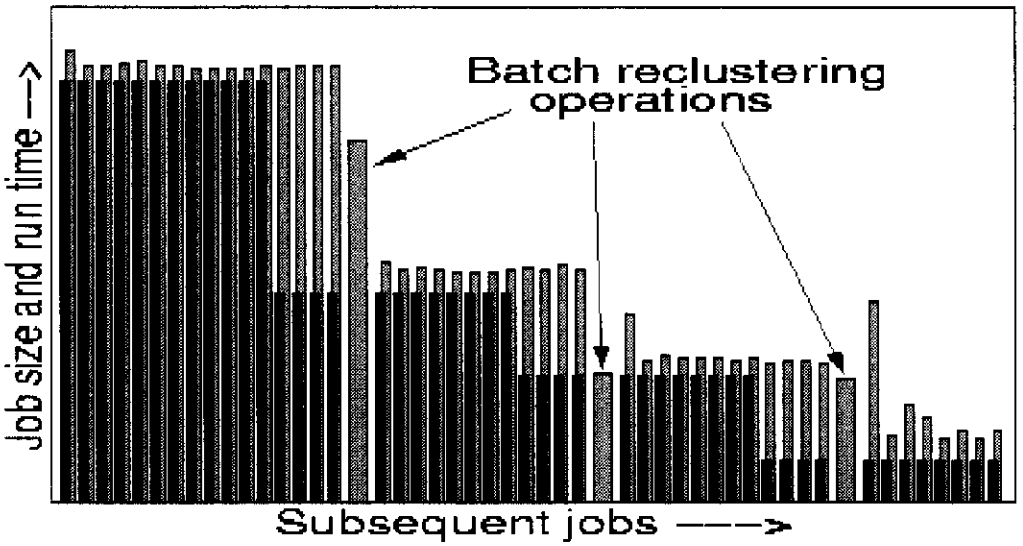

Figure 23 - Job time for accessing clustered versus unclustered data

Clustering and re-clustering has also been studied within the ATLAS collaboration [37], where a so-called Hamming algorithm has been developed, as it is based on the use of the Hamming distance between two bit-vectors. Using this algorithm, data are clustered according to multiple access patterns, without any data duplication. By controlling the order in which data are accessed (the iteration order), this algorithm reduces the number of disk seeks to almost the theoretical limit. Performance is maintained for 15 – 40 access patterns, depending on the overall selectivity. For larger numbers of access patterns, the use of data duplication becomes attractive.

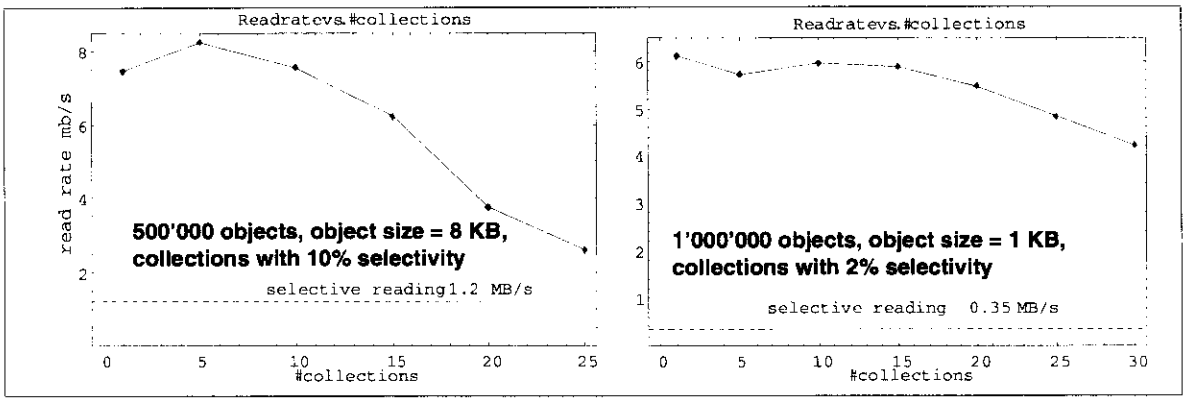

Figure 24 - Disk Re-clustering using the Hamming Algorithm

Tape clustering has been studied as part of the HENP Grand Challenge Project in the US. In this case, data are clustered on tape according to the distribution of events in a multi-dimensional space. This type of clustering has also been studied in the NA48 collaboration, and is suitable for cases when the number of dimensions is relatively small.

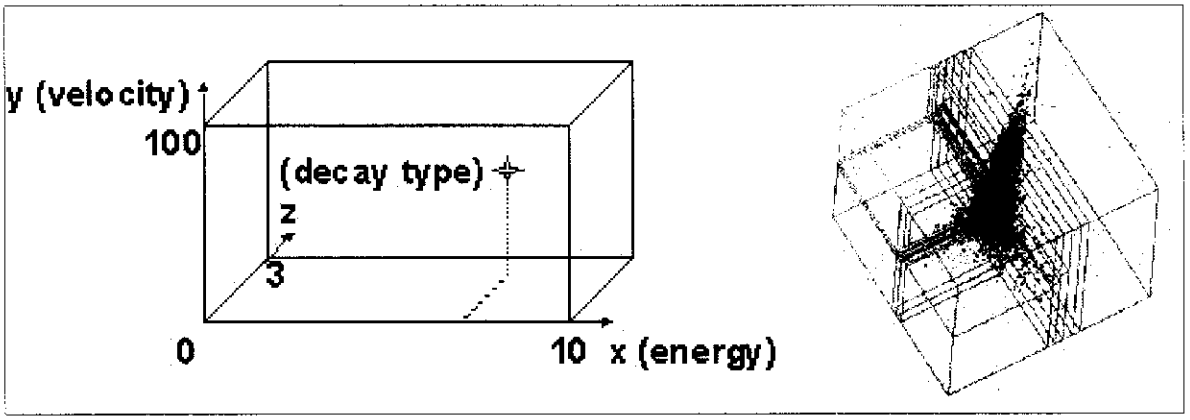

Figure 25 – Distribution of Events in a multi-dimensional Space

Other tape-related clustering issues that have been studied, in CMS, include filtering and clustering of data chunks cached from tape (see Figure 26).

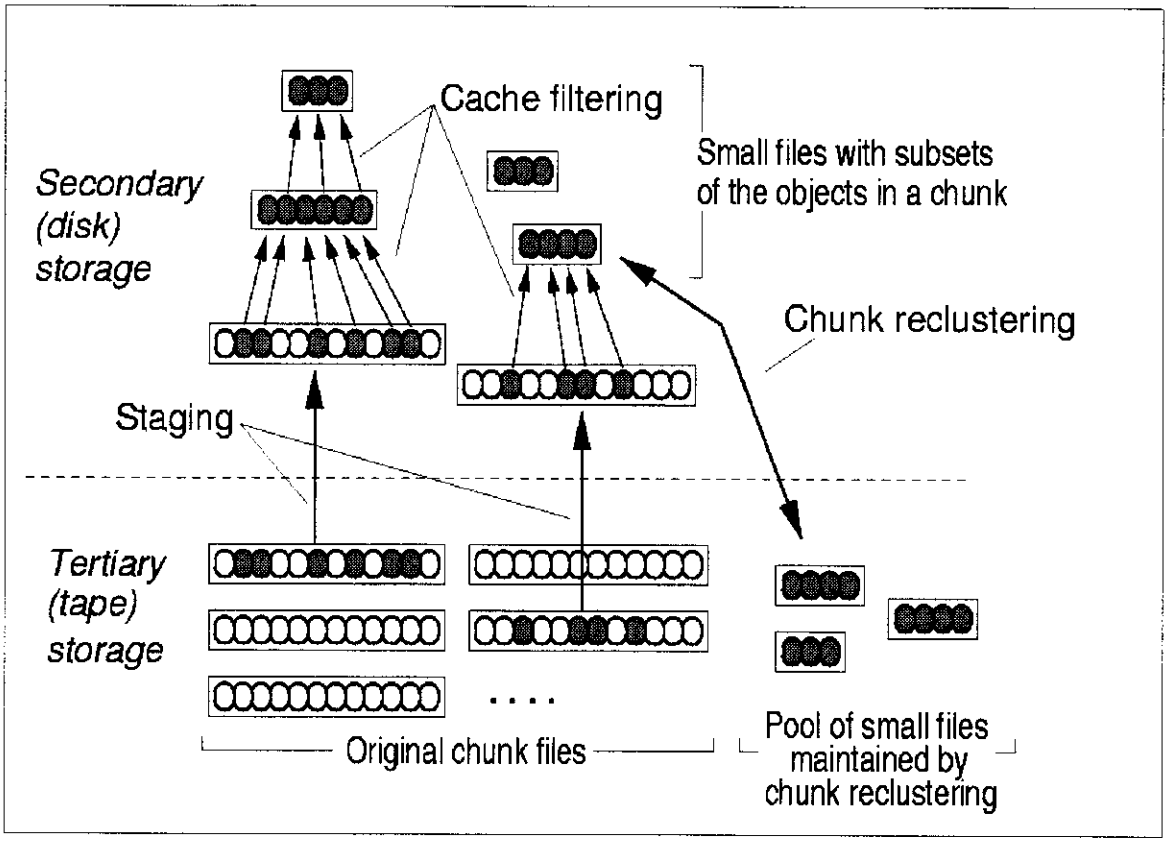

Figure 26 - Cache Filtering and Chunk Reclustering

In conclusion, it appears that good clustering and re-clustering will be important to achieve and maintain good I/O performance. The effectiveness of any of the re-clustering strategies studied above is strongly coupled to user access patterns, which are not yet well known, and to the performance and characteristics of the storage devices that will be deployed.

7.3 Mass Storage Integration

Although the architecture of Objectivity/DB theoretically permits federations in excess of 1EB (1000PB)[15], it is not currently feasible to store such a large quantity of data on disk, as is assumed by the basic Objectivity/DB architecture. Hence, a mechanism whereby inactive databases can be stored offline – e.g. in a Mass Storage System (MSS) – is required. An interface between Objectivity/DB and HPSS was agreed at a joint meeting between representatives of the major HEP laboratories, Objectivity and the HPSS consortium in May 1997. The implementation of the interface has been performed as a collaborative effort between Objectivity, SLAC and CERN.

A prototype of an interface Objectivity/DB and HPSS was presented in a previous RD45 Status Report to the LCB [1]. This prototype interface suffered from performance problems, due to the mismatch between HPSS – tuned to perform well on large data

[15] In addition to the disk space requirements, federations of such a size would require enormous files, as is discussed in section 12.5.

transfers – and Objectivity/DB, where small data transfers are used. To circumvent these problems, it was proposed that the original interface, which used direct calls to the HPSS client API, be replaced by a more traditional staging system. Although such a system would not exploit the disk-cache management capabilities of HPSS, it would enable the performance problems of the prototype system to be avoided.

The interface between the two systems is shown schematically in Figure 27, and consists of 3 components.

1. A linkable version of the Objectivity/DB server – or AMS – where all I/O calls conform to a standard interface.
2. A back-end storage system.
3. Storage-system dependent interface code.

The use of the clean I/O interface enables alternative MSSs to be used, as has been demonstrated at INFN/Rome. This is important not just for smaller sites, which cannot necessarily afford or do not need the functionality and complexity of a system such as HPSS, but also to allow a smooth migration path to alternative systems in the future.

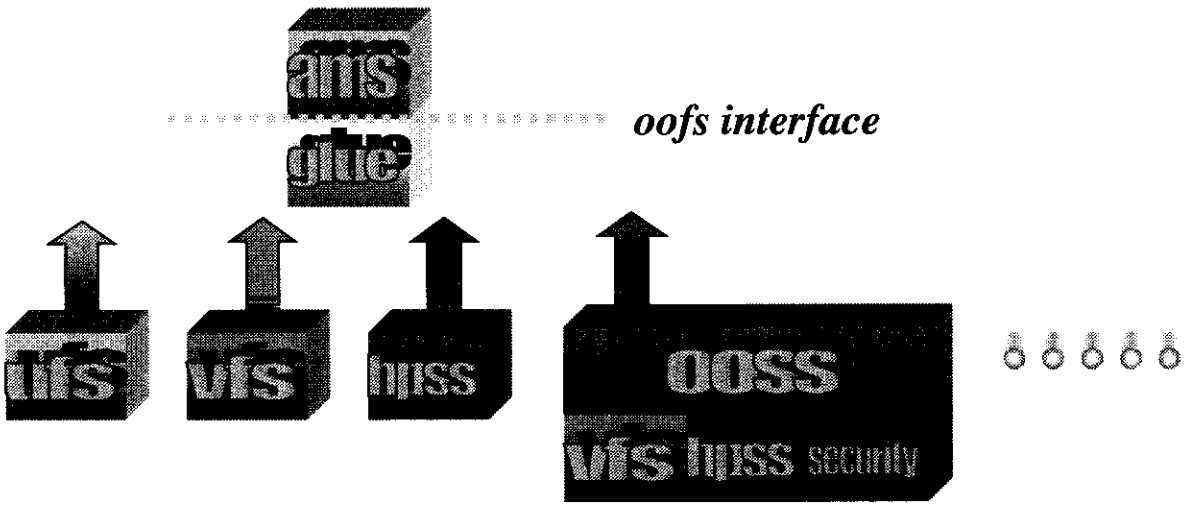

Figure 27 - Objectivity/DB - HPSS Interface

When an attempt is made to access a database that is already present in the stage pool (see Figure 28), the operation procedes normally – as if the standard AMS were used. However, if the database in question is tape-resident, a stage request is generated and the open operation is blocked until the database is staged to disk. In addition, a free-space daemon assures that sufficient disk space remains available, migrating files to tape, using a least-recently-used algorithm, as required.

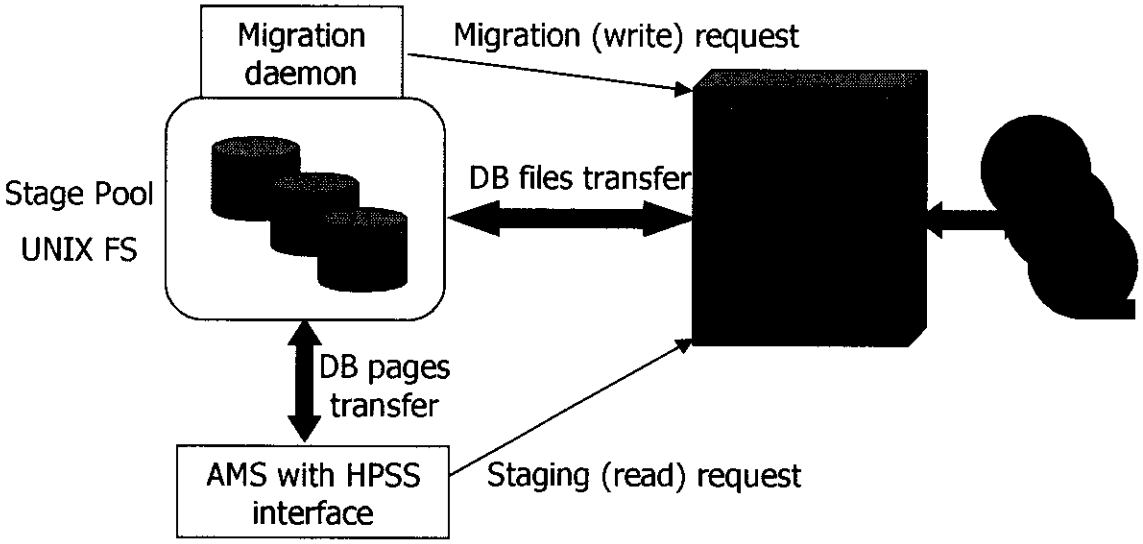

Figure 28 - Staging Interface

The interface between Objectivity/DB and HPSS is used in production at both CERN and SLAC, but with slightly different configurations. At SLAC, PFTP is used to move the data between HPSS and the stage cache (see Figure 29 and Figure 30), whereas at CERN, the SHIFT RFIO package is used. At CERN, over 1TB of data have been stored using the interface (primarily the ATLAS 1TB milestone, described in section 4.3 on page 18), whereas at SLAC over 1TB of production data have been stored.

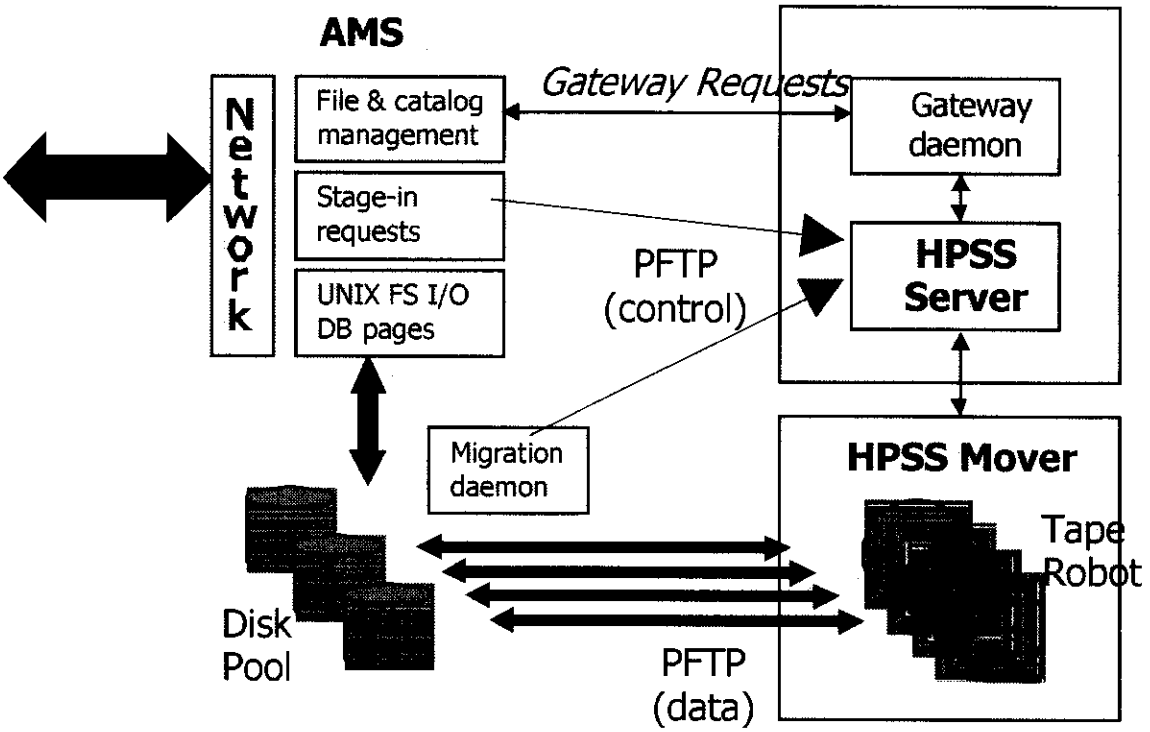

Figure 29 - SLAC Configuration with Remote Tape Access

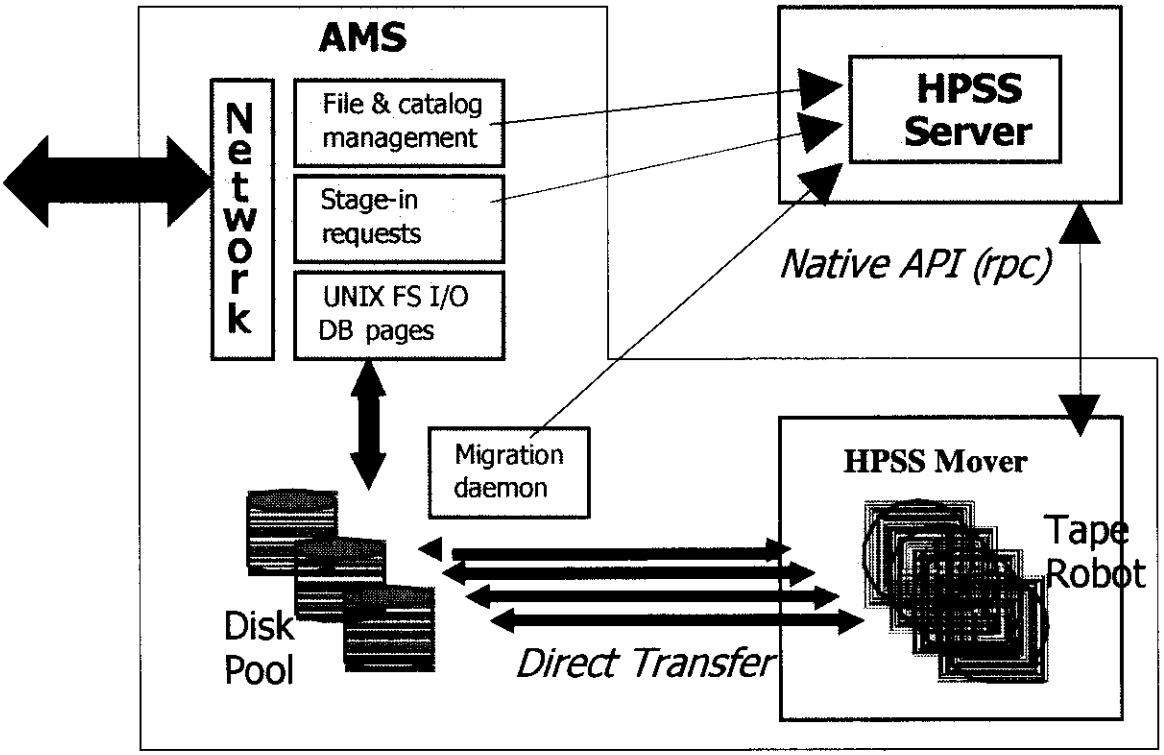

Figure 30 - SLAC Configuration with Local Tape Access

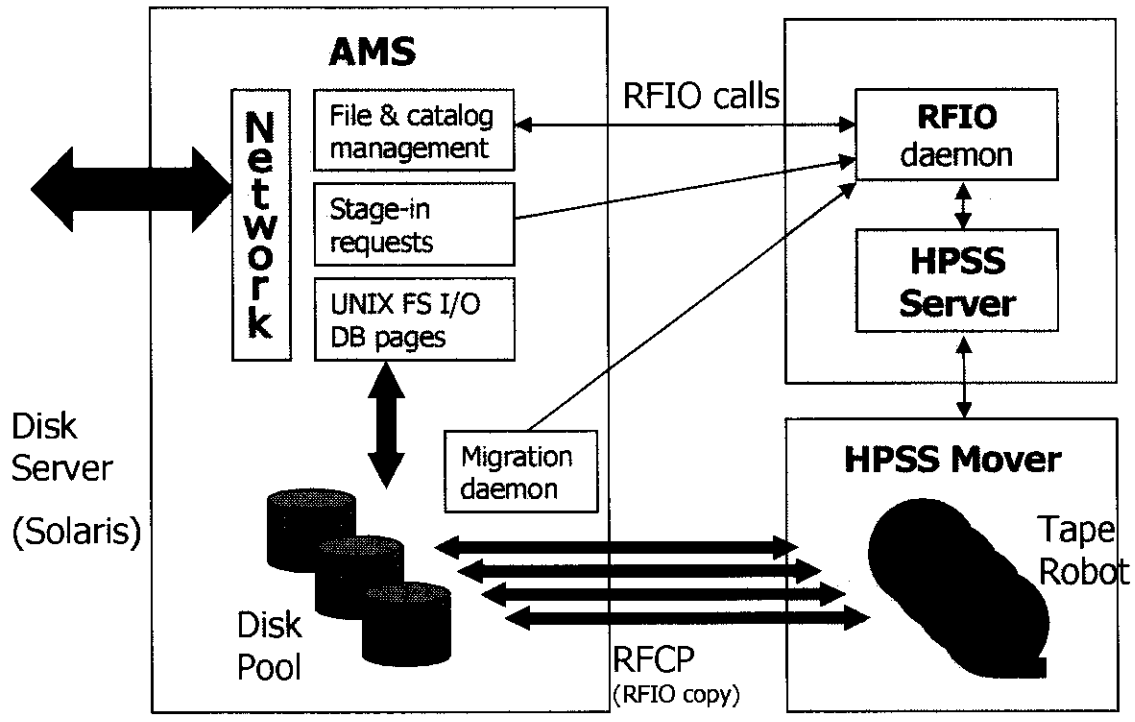

Figure 31 - CERN Configuration

At the time of writing, both CERN and SLAC use Objectivity/DB V5.1.x. This version does not contain the official production release of the interface. In particular, the V5.1 AMS will

block until an I/O operation is completed – clearly a problem in the case of offline databases, as the operation may take several minutes to complete. Although it is possible to increase the RPC timeout between client and server, a much cleaner solution will be provided in Objectivity/DB V5.2.

In this version, the AMS will be able to tell the client to retry after a specified time interval, thus avoiding timeouts in the case of lengthy operations. This mechanism is shown schematically in Figure 32.

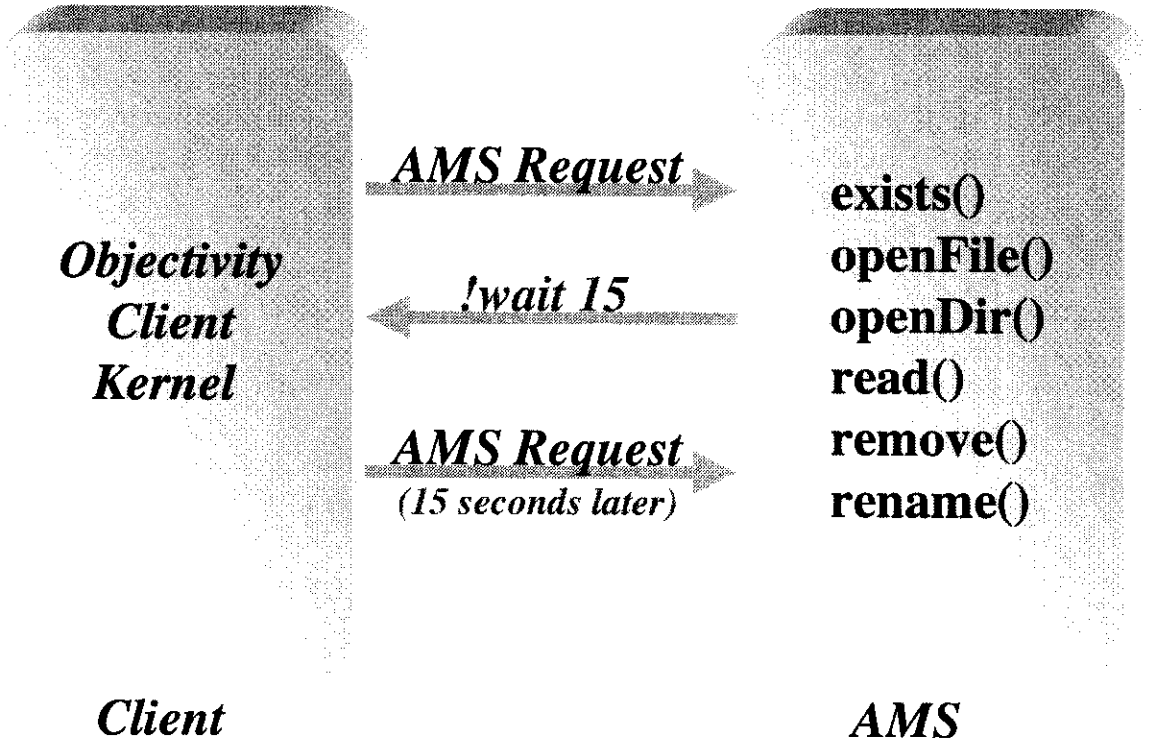

Figure 32 - AMS Defer Protocol

Clearly, further work will be required to integrate the production release of Objectivity/DB V5.2 when it is released. In addition, the numerous other enhancements designed by SLAC and scheduled for this release will have to be tested and put into production. However, these activities are clearly more related to production rather than R&D. Hence, the issue of Mass Storage Integration is now considered one of production.

7.4 Multi-User, Multi-Federation Issues

7.4.1 Introduction

Although it was originally foreseen that a given experiment would use a single Objectivity/DB federation, it soon became clear that multiple federations would be preferable. The revised scheme involved a single *reference federation* – containing only the schema – a *production federation* – containing the schema and data, as well as multiple *developer federations*, created by cloning the reference federation and attaching copies of the production databases as required. Experience with such a scheme has shown that the use of multiple federations can solve a number of problems, described in detail below.

Multiple consistent federations are a solution to the following problems:

- Decoupling
  - Unwanted lock contention, e.g. between online and offline systems, is avoided.
  - Unwanted changes to the database schema or catalog are avoided.
- Distribution and Backup
  - A partial or complete copy of a federation can be safely transferred to remote machines in the case of slow or unreliable networks, that prevent the use of a single distributed federation.
  - A backup of the central meta-data – again a partial copy – can be made.

These issues could also be addressed by enhancements to Objectivity/DB, such as support for private schema and data, partial backup of a federation and so forth. It is hoped that these issues will be addressed with time, reducing the need for what are essentially work-arounds.

Examples of the above cases are given below.

7.4.2 Decoupling Environments

It would be clearly highly undesirable for the online data acquisition system to be blocked by a single read lock taken by a user who had opened a database browser. To avoid such problems – which would occur frequently during periods of intensive offline analysis – a simple solution is to deploy two federations. In such a scenario, the "offline" federation would be created as a copy of the "online" federation and both would run independently with separate lock servers and catalogs. On a periodic basis, databases would be copied from the online federation to the offline one, or attached to both federations. In the latter situation, it is clearly necessary that updates are only performed from one side, or preferably not at all.

The use of multiple decoupled federations is greatly simplified if the data flow is uni-directional. A further simplification is if databases are populated and then become readonly. This is typically not the case for databases containing metadata, which typically have to be recopied from the source to target federation.

7.4.3 Decoupling Schema and Catalog

Another frequently encountered situation is the need to share the bulk data amongst different users, each of whom requires additional private or semi-private databases and schema. Such a case can be handled by using multiple cloned federations, where each user who needs this capability starts with a clone of the production federation – a copy of the complete catalogue and schema. The production databases are visible to both the original production federation and the clone, as they are referenced by both catalogues. The clones could even use the same lockserver and federation ID as the main production database. A user could then add new schema to their private clone, and new databases and instances of their private classes (or the production classes). Naturally, these changes would not be visible to the production federation.

Such a scheme would also permit databases to be shared between different users, as is shown in Figure 33. Clearly, such a scheme would have to be used with care, and would require the separation of users' schema into different named schema. It is also somewhat inefficient, as it results in multiple copies of the master federation catalogue, with rather small differences, resulting from the end-user schema and databases.

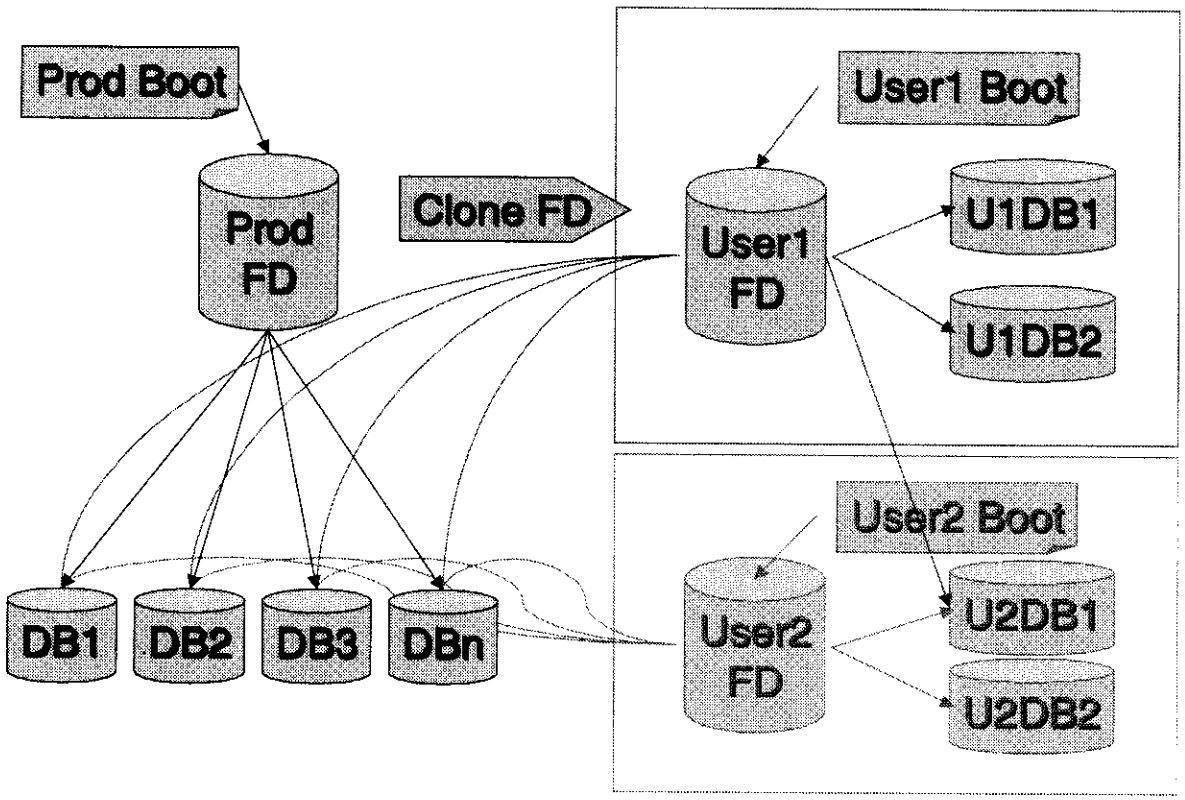

Figure 33 - Using Multiple Cloned Federations

7.4.4 Copying a Federation

The recommended tools for making a copy of an Objectivity/DB federated database are *oocopyfd* or *ooinstallfd*. The use of *ooattachdb* is not recommended, except for the case when the database ID should be changed on attach.

7.4.5 Backing Up a Federation

In the typical HEP environment, a database backup normally applies to a small subset of the federation – it makes no sense to stage in and recopy to tape readonly databases resident in HPSS. The important data to backup is essentially meta-data – the schema and catalogue and "registry" information, such as the naming tree, collections (of references – not data!) and so forth. As the current Objectivity/DB backup tools (*oobackup/oorestore*) do not support partial backup, an alternative solution has to be found. A simple technique is to make a partial copy of the federation, perhaps to a special disk area, and then dump this copy to tape. A Perl script to perform such an operation is provided in the HepODBMS tree.

7.4.6 Summary

The use of multiple federations provides a pragmatic solution to some limitations in the current version of Objectivity/DB. The techniques described above are used in production by a number of experiments, including both BaBar and CMS. White papers describing these techniques, together with appropriate scripts and/or other tools and documentation, are in preparation.

7.5 Conclusions

Extensive work on the R&D items requested by the LCB has been carried out. Much of this work has been performed within various experiments and/or the MONARC project, underlining the increasing involvement of the collaborations in these activities. Where appropriate, recommendations, tools and associated documentation has been or will be developed and will be made available as part of the RD45's production activities.

8 Experience at BaBar

8.1 Introduction

The BaBar experiment at SLAC is designed to perform high-precision studies of the decays of the B-meson, that are produced in e^+e^- collisions at the PEP-II accelerator. Some 650 scientists from 85 institutes in 10 countries are members of the collaboration. The first physics data was taken on May 26th 1999. The overall characteristics of the experiment are shown in Table 4.

Characteristic	Size
Number of Detector Sub-systems	7
Number of electronic channels	~250,000
Raw Event Size	~32Kbytes
DAQ to Level 3 Trigger	50MB/sec (2000Hz)
Level 3 to Reconstruction	2.5MB/sec (100Hz)
Reconstruction	7.5MB/sec (100Hz)
Event Rate	10^9 events/year
Storage Requirements (real + simulated data)	~300TB/year

Table 4 - BaBar Characteristics

8.2 Database Requirements

The overall requirements on the database system are as follows:

- Provide storage and access for event data (Event Store):
  - Handle >10^9 events/years, >100KB/event.
  - ~300TB/year of real and simulated data.
- Handle the detector conditions data (Conditions Database):
  - Track detector performance as a function of time.
- Support distribution and data access across the entire collaboration.
  - Both WAN and LAN access.
- Other miscellaneous databases.
  - e.g. production database

In addition, the system must be capable of handling changing requirements, such as new physics goals, modifications to the detector hardware and so forth.

8.3 Overall Design

The overall database design consists of a number of *domains*, such as the event store, conditions, etc., which are logically independent, even if physically coupled through the use of the underlying technology, namely Objectivity/DB. The use of independent domains was important in that it allowed parallel development of the various domains, which were then brought together fairly late on in the development cycle. The *EventStore* domain is given the responsibility of being the primary transaction manager. Client-side multi-threading is not exploited, as it was not supported by the database at the time that the overall design was frozen. On the other hand, multiple database contexts (*ooContext)* are used to handle "mini-transactions".

8.4 Event Structure and Placement

The event structure and average size of each component is shown in Table 5.

Data Type	Name	Size per event (KB)
Simulated Data	SIM	~100
Truth	TRU	~40
Raw Data	RAW	~30
Reconstructed Data	REC	~100
Event Summary Data	ESB	~20
Analysis Object Data	AOD	~2
Event Selection Tag	TAG	~.2 (200bytes / event)

Table 5 - Data Types and Placement Regions

In order to optimise access, data are clustered according to the event component as shown in Figure 34. The clustering algorithm automatically rolls over to a new container / database / filesystem as required and transparent navigation to the event data is provided by the appropriate headers. Named event collections – which may contain both pointers to events and/or other collections – are provided, using a Unix-like naming tree.

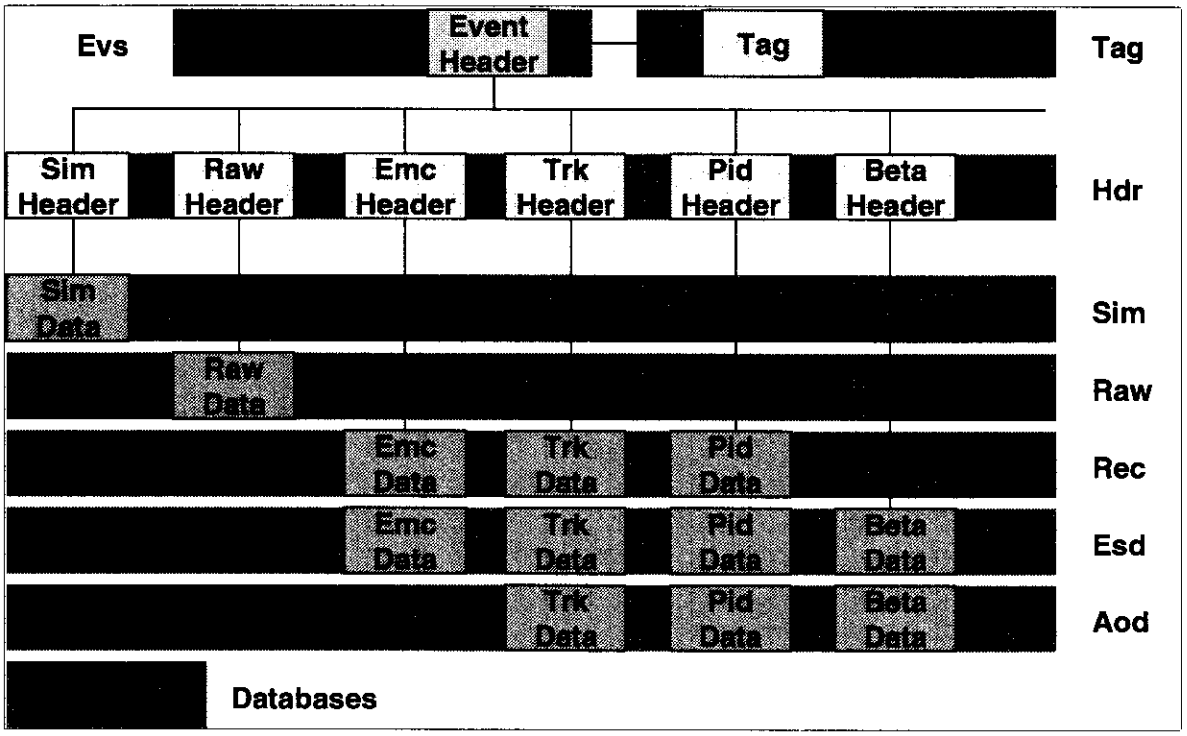

Figure 34 - Event Structure and Placement

8.5 Processing Framework

Reconstruction and analysis uses the same processing framework (see Figure 35). An *input module* creates a skeleton event from the input source. Different input modules are used to cater for different sources. *User modules* may be grouped together into sequences and multiple paths are supported. A *filter module* can terminate the processing of a given path, which automatically initiates the processing of the next path. User modules see only those events that are passed down the path in question. *Output modules* provide an abstraction of an output stream, which corresponds to a set of databases or named collection, such as */groups/multihadron/taus/theTauSample*.

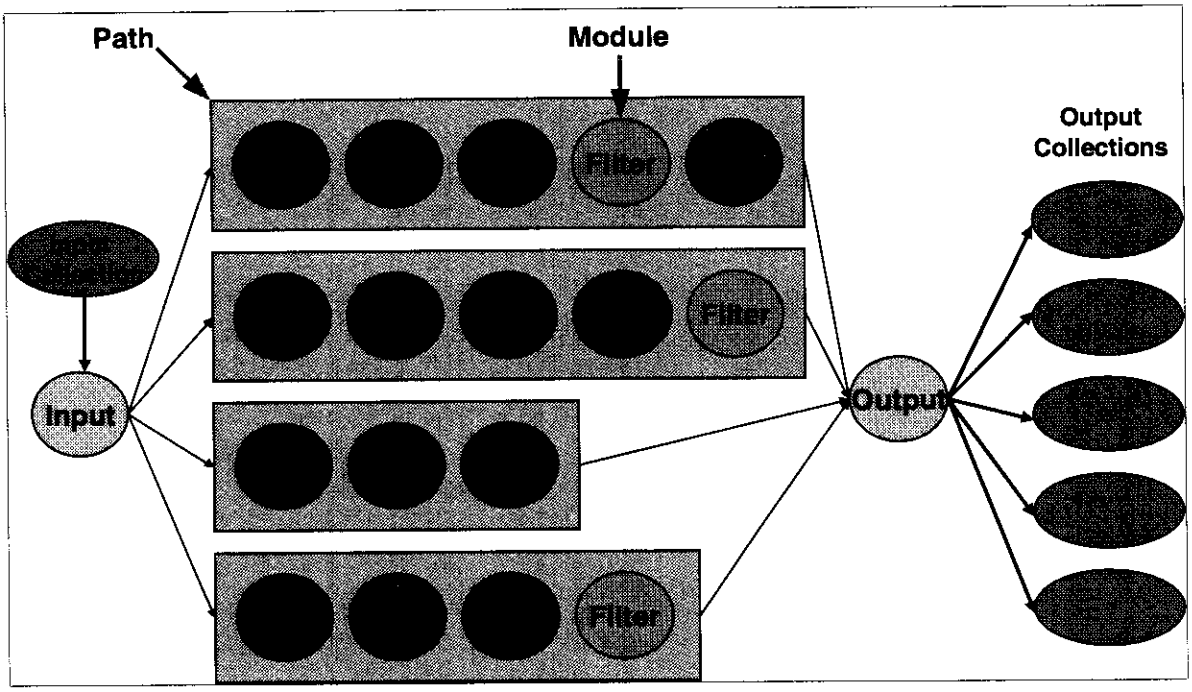

Figure 35 - BaBar Processing Framework

8.6 Transient / Persistent Decoupling

In the BaBar model, user code deals only with transient objects. Input modules are used to create an empty transient event and loader modules convert parts of the persistent event to the transient representation. In a similar manner, convertors are used to generate persistent data from the transient information on output. This decoupling hides the database implementation from the user, which would in principle allow for a replacement of the underlying store. It allows the data to be represented differently in the two worlds – for example, small transient objects can be grouped into a single persistent object to minimise the storage overhead associated with such objects.

8.7 Release Builds

During the development phase, full releases are built every 2 weeks on both Digital Unix and Sun Solaris are currently supported. Linux will shortly be added to the list of supported platforms, which previously also included HP/UX and IBM AIX. Between the fortnightly full releases, 1-2 bug fix rebuilds and nightly "compile only" rebuilds occur. For each build, a new federation is created, with the schema built from scratch. The schema building is performed on a single platform and then exported to the others. Developers and users import and initialise test federations from the exported release federation for a given release. Code is then developed against these test federations. Schema evolution is permitted during this phase and schema changes become official with the next release build. The source code, managed using CVS, is organised into some 550 packages with a total of 5800 classes. Packages are tagged by a version identifier and releases are defined as being a set of tagged packages.

In the production phase, a reference federation, containing only schema, is used. Once the schema associated with a given release have been validated, these release schema become the new reference schema. In addition to the reference federation, there are a number of production federations, containing both schema and data. The production schema are upgraded, currently on a weekly basis, from the reference schema.

8.8 Schema Evolution

It was initially intended that the schema evolution feature provided by Objectivity/DB would be used. However, as this feature normally requires that existing executables must be rebuilt, which creates problems in reproducing existing analyses, it was decided to move to explicit schema migration, where the schema are versioned "by hand". The transient – persistent mapping layer is used to perform the appropriate translation.

8.9 Data Distribution

Regional centres for BaBar existing in France (IN2P3), Italy (INFN) and the UK (RAL) with SLAC acting as a regional centre for North America. Bulk data transfer between these sites is performed using database files. For data exchange between developers, small event samples are copied, which is more efficient than exchanging complete databases. A large suite of tools, based on Perl and C++, to support the export of collections, domains and so forth is being developed. Although the Objectivity/DB FTO and DRO features are not currently used, it is intended to re-evaluate these tools at a later stage.

8.10 Data Protection and Safety

There is a clear need to provide some level of protection to the data. This is provided both within a federation – e.g. to protect against accidental corruption or deletion – and at the level of the administration tools, such as *oodeletefd*. Within a federation, authorization levels are used, supporting the concept of *system*, *group*, and *user*. The tools themselves are wrapped to prevent accidental misuse, although this is not ideal as the original utilities can still be found. A longer term solution is being provided as part of the joint developments between SLAC and Objectivity, which will include security hooks on both client and server sides, allowing a site to plug in its own authorisation mechanism.

8.11 Access Statistics

In order to generate access statistics, a post-processor is used to insert code into that generated by the DDL processor. This allows statistics to be obtained on which databases and objects are accessed. It is possible to enable / disable this feature at run-time and a non-measurable overhead is incurred when disabled.

8.12 SLAC Configuration

The SLAC configuration, shown in Figure 36, is primarily based on Solaris, although Linux systems are planned. 2 SMPs are provided for software development and physics analysis, where 64 CPUs will shortly be available for the latter. 8 datamovers – providing the disk cache and HPSS interface – are currently used, together with a 200 node processing farm and 100 node online system. The disk caches on the datamovers consist of various regions, including:

- Staged (migrated/staged/purged).
- Resident (migrated).
- Dynamic (not migrated).
- Test (not managed).

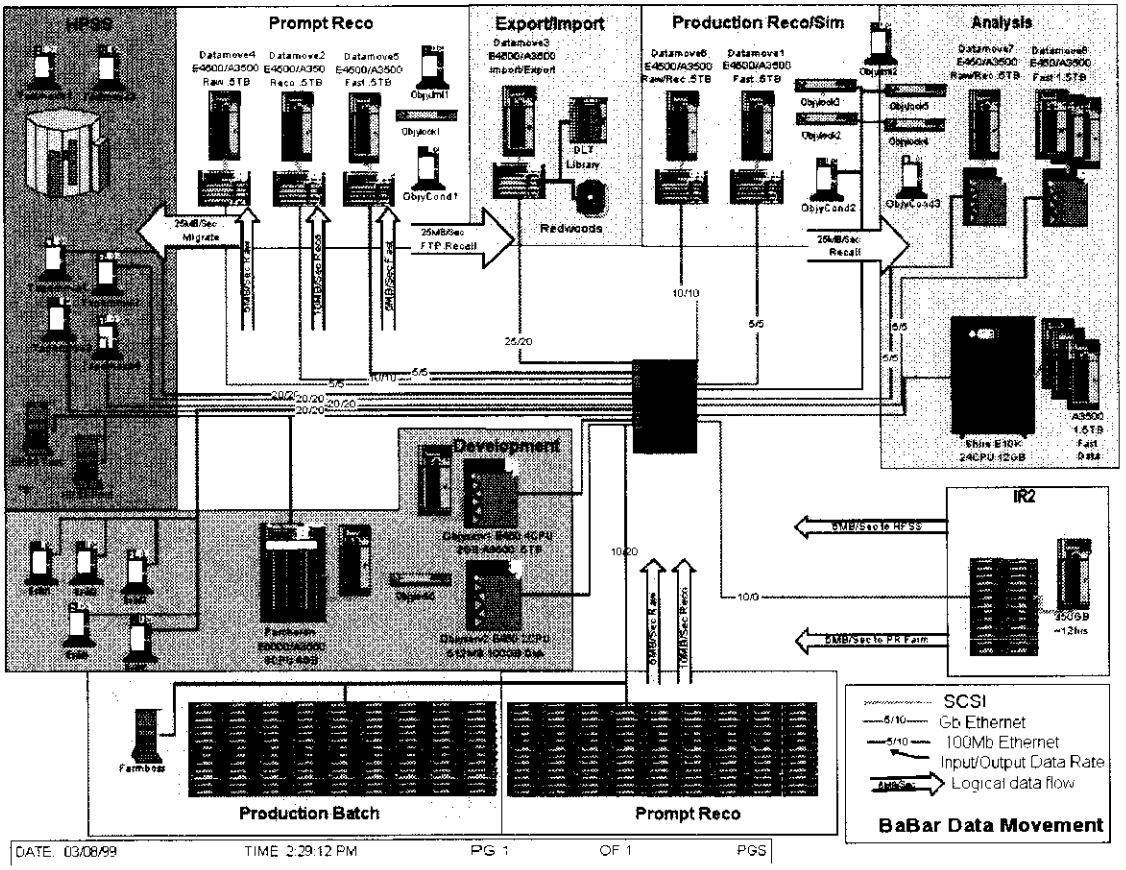

Figure 36 - SLAC Hardware Configuration

8.13 Production Federations

Multiple production federations are currently deployed, including:

- Physics.
  - Online.

    - Deadtime critical.
    - Uses FTO and soon DRO.
  - Analysis.
  - Reprocessing.
    - For bulk reprocessing with improved algorithms / alignments etc.
- Simulation.
  - Generation.
    - Production loading of simulated data.
  - Analysis.
  - Reprocessing.

Data is moved between production federations using a mixture of data distribution (physical copying of databases between federations using a DBID allocation scheme) and shadowing, where databases are attached to multiple federations once readonly.

8.14 Database Commissioning and Performance

At the time of the RD45 workshop in July 1999, BaBar production (event reconstruction) was running at the rate of some 10-15 events/s (with peaks of 50Hz) on 60-70 nodes (up to 150 nodes observed). The target is to reach 100Hz by February 2000 – the rate at which events are acquired from the detector – on some 200 nodes without sacrificing robustness. We describe below on-going work on performance optimisation of the reconstruction farm.

In addition to this work – and perhaps more important in the eye of the end-user – the performance of the analysis environment needs to be improved. This latter activity was only just beginning at the time of writing and hence cannot be described in this report. However, details will be distributed via the normal RD45 mailing lists as soon as they become available.

Performance optimisation is clearly an on-going process. Since the start of BaBar data taking in May, a number of successful steps have been carried out, as follows:

- The previously long startup time (see Figure 37) has been reduced from 20 to 3 minutes.
- Clustering hint initialisation redesigned, using a lock-free strategy.
- 3 bugs in Objectivity/DB solved rapidly with the cooperation of Objectivity.
- Optimisation of container naming strategy (40% improvement).
- Optimisation of access to Objectivity/DB catalogue.

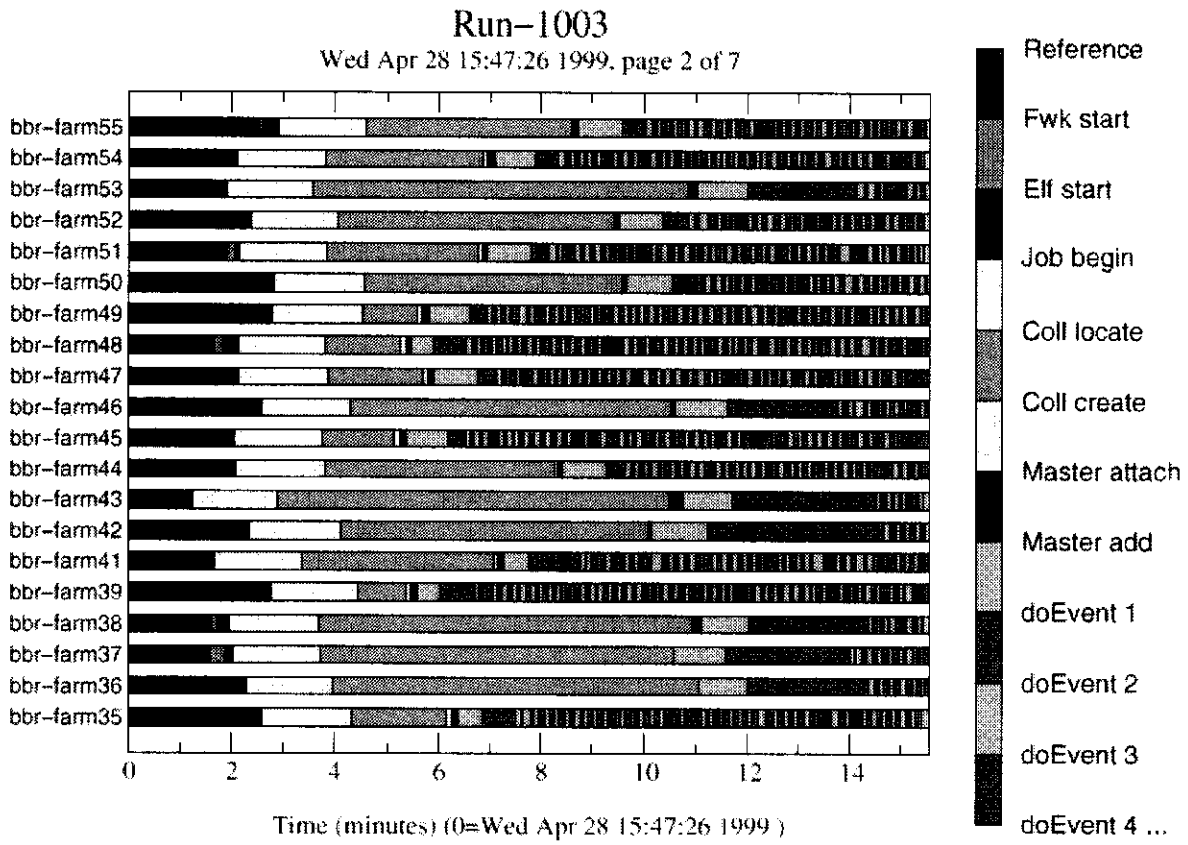

Figure 37 - BaBar Job Startup Time

Additional optimisations that have been performed include the following:

- Tuning transaction granularity.
  - Randomization.
- Tuning initial container size and container growth.
  - could improve later access (analysis).
- Multiple AMSes per host.
  - One AMS is started per CPU per data server (4-CPU Suns).
- Database clusters.
  - Independent sets (clusters) of databases written to by corresponding subsets of the reconstruction farm.

As a result of these steps – made in a single week after the appropriate hardware had been installed – the overall throughput was improved by some 224% (see Figure 38).

No saturation in performance is seen up to 110 nodes, although at the time of writing (15th September) a small degradation – less than 10% at 150 nodes – is still seen if more nodes are used.

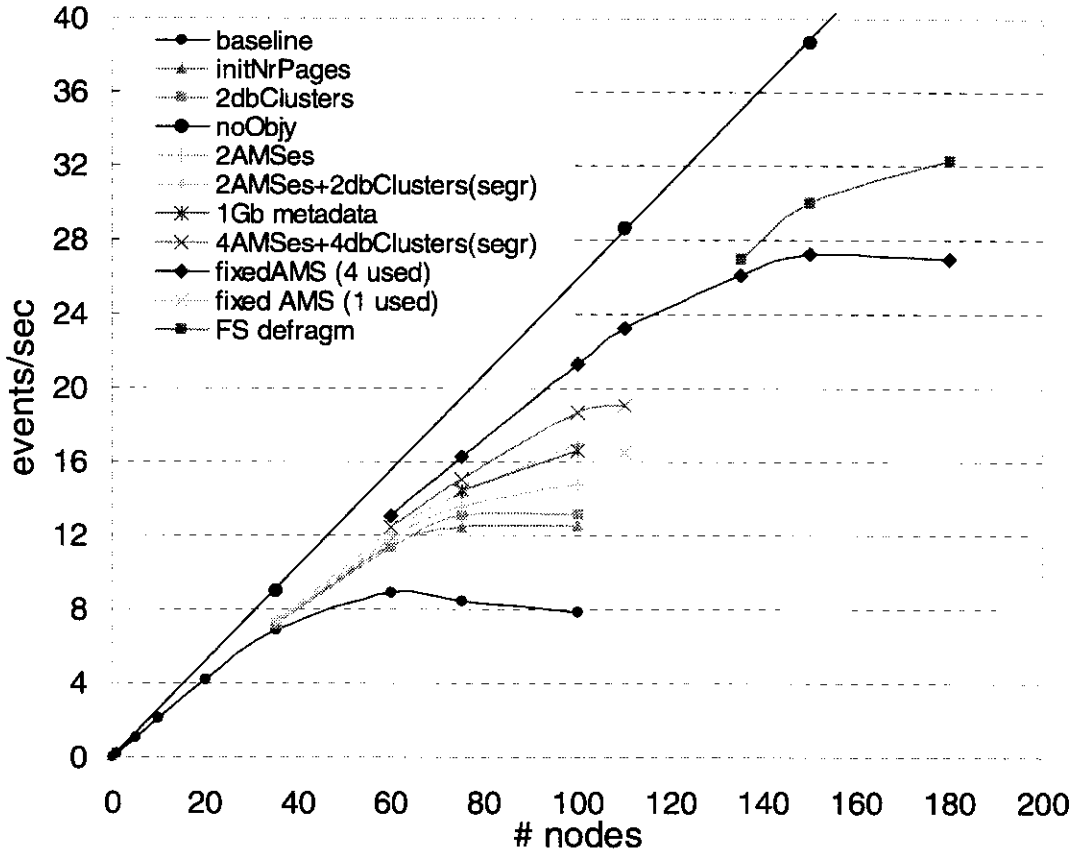

Figure 38 - BaBar Reconstruction Throughput

A number of other optimisations are in progress, namely:

- Tuning of the Objectivity/DB cache.
  - Ongoing work with Objectivity engineers.
- Ongoing work on the "logging manager" to permit longer transactions.
  - The logging manager reads raw events and distributes them via CORBA to the reconstruction processes. All events are kept in memory until processed successfully and committed.
- Process data in larger chunks.
  - Each run currently processes a 2GB input file. As the startup overhead is significant, processing larger chunks will improve the overall performance.
- Database pre-creation.
  - Factor out the database creation time from the reconstruction process.
- The use of multiple autonomous partitions (to reduce the load per lockserver).

Possible further steps include:

- Doubling the processing stream.
  - e.g. 2 PromptReco federations.

- Further improvement in accessing shared resources.

8.15 Current Status

Number of Sites Using Objectivity/DB	>15 (USA, UK, France, Italy)
People who have signed license agreement	~500
People who created a test federation	~260
Simultaneous users (oolockmon output)	80-90
Developers (created or modified a persistent class)	~30 (10 "experts")
Number of Persistent Classses	450

Table 6 - BaBar Database Usage Statistics

8.16 Summary

The BaBar database system, based upon Objectivity/DB and HPSS, is now in production and has already written several tens of TB of data. The basic concepts of an ODBMS have been shown to be a good match to the problem, although a number of issues, primarily related to performance, scalability and robustness, still need to be addressed. A schedule is in place to address these concerns and other problems that will no doubt arise as the data volume and usage builds up. At the time of writing, significant improvements in the overall performance of the reconstruction federation had been achieved, although additional enhancements are still required. Attempts to optimise the performance of the analysis federation were only just starting and hence cannot be covered in this report. However, it is expected that some of the techniques used on the reconstruction federation will also be of benefit on the analysis side. Progress on both reconstruction and analysis federations will be closely monitored and reported on via the RD45 mailing lists and Web pages.

9 Experience at DESY

9.1 H1

The H1 experiment at DESY has recently implemented a tag database built on Objectivity/DB. In the current – Fortran-based – system, event selection is performed using index files, based on a 32-bit classification word. The tag database, on the other hand, contains:

- Kinematic and topological variables of each event.
- Major final state objects.
- Control variables (trigger, filter, ...)

The basic aim of moving to a tag database is to speed up data access by providing improved event selection possibilities. Interfaces to the tag database are provided for C++, Java and Fortran, including a Objectivity/DB Data Interface Module (DIM) for Java Analysis Studio (JAS).

Further details regarding the H1 tag database can be found via http://www-h1.desy.de/~benisch/talks/rd45_990712.ps.gz.

9.2 ZEUS

The ZEUS experiment has built a similar system to the above, replacing their existing ADAMO-based system [23]. The Objectivity/DB-based system – ZES – has been in operation since late 1997. Information stored in ZES includes:

- Run and event number.
- $1^{st}/2^{nd}/3^{rd}$ level triggers and DST bits.
- Various other bits and flags (good run, beam type, pattern recognition flags, ...).
- Detectors.
  - Calorimetry, tracking, etc.
  - Taggers.
  - Luminosity monitor.
- Pattern recognition.
  - Electron finders.
  - Jet finders.
  - Track finders, track matching.

The ZES federated database currently consists of some 350 database, each of which are between 200 and 400 MB in size. Each database contains about 10 run objects, which in

turn consists of some 10K event objects. For each event, the DAQ information, some 250 physics and detector variables and a pointer to the full event in ADAMO is stored.

Year	ZES version	# events	Data Volume	Per Event
1995	V1.0	7.9×10^6	12.8GB	0.7KB
1996	V2.0	20×10^6	48.3GB	0.9KB
1997	V2.0	30.4×10^6		
1998	V2.0	9.5×10^6	10.6GB	1.1KB
Total		77.8×10^6	71.7GB	0.9KB

Figure 39 - ZES Storage Requirements

ZES is integrated into the standard ZEUS analysis environment, for which there are typically between 0 – 100 simultaneous users (normally around 20). Typically, a user reads between 10^2 and 10^3 runs, corresponding to some 10M events.

10 RD45 "White Papers"

Since the time of the last LCB review, a series of topical "white papers" has been produced[16]. The main purpose of these papers is to provide information in a concise and easily digestible form in a more timely manner than is possible via LCB status reports.

A number of these documents concern guidelines and recommendations for production deployment of Objectivity/DB. Others document work on the use of Java agents and a database administration tool written in that language. The bulk of the remainder is concerned with the on-going RD45 risk analysis and issues arising.

A list of titles is given below. In addition, the revised risk analysis report is reproduced in its entirety in chapter 11 on page 72.

- Objectivity Testbed Parameterisation, CERN/RD45/99/12, June 1999
- Interim Status Report, CERN/RD45/99/11, June 1999
- Intelligent Histogram Storage, CERN/RD45/99/10, June 1999
- ODBMS Manpower Requirements, CERN/RD45/99/09, March 1999
- Persistent Object Manager Selection, CERN/RD45/99/08, March 1999
- ODBMS Experience, CERN/RD45/99/07, March 1999
- A HEP ODBMS Testbed, CERN/RD45/99/06, March 1999
- Risk Analysis Update, CERN/RD45/99/05, March 1999
- Objectivity/DB Evolution, CERN/RD45/99/04, February 1999
- Requirements for Multi-File DB Support in Objectivity/DB, CERN/RD45/99/03, February 1999
- Requirements for the Objectivity/DB - HPSS Interface, CERN/RD45/99/02, February 1999
- Persistent Object Manager Choices, CERN/RD45/99/01, January 1999
- Mobile Agents in Java, CERN/RD45/98/12, December 1998
- Database Administration Tool (DRO_TOOL), CERN/RD45/98/09, November 1998
- Risk Analysis and Management, CERN/RD45/98/08, October 1998
- Computing Models and Reliability Overheads, CERN/RD45/98/07, August 1998
- Wide-Area Objectivity/DB Federations, CERN/RD45/98/05, July 1998
- Allocation of Federated Database Identifiers in Objectivity/DB, CERN/RD45/98/04, July 1998
- Guidelines for the use of Replication in Objectivity/DB, CERN/RD45/98/03, July 1998
- Guidelines for Setup of Production Objectivity/DB Federations, CERN/RD45/98/02, June 1998
- Assumptions concerning Distributed Data Management, CERN/RD45/98/01, May 1998

[16] See http://wwwinfo.cern.ch/asd/rd45/recommendations.htm.

11 Risk Analysis

The following is a direct copy of the RD45 "Risk Analysis Update" - CERN/RD45/99/05.

11.1 Introduction

This document is essentially an annotated version of "Risk Analysis and Management" CERN/RD45/98/08. In that document, proposed actions were written in red. Comments on or updates to these actions appear in green, as this text.

In this document we attempt to identify the key risks involved in the strategy for HEP data management that is currently proposed by the RD45 collaboration. In each of the key areas, we make proposals for future work aimed at minimising, or at least developing a better understanding of, the corresponding risk factor.

We emphasise that risk analysis and management is an on-going effort and the issues described in this document build on earlier RD45 work, as presented to the LCB, LCRB and DRDC.

In addition, this document is intended to serve as the basis for discussion. The action items listed below are merely suggestions – the concrete list of actions will be drawn up as a result of the discussions held with the experiments and the LCB[17].

11.2 HEP Computing Models

It is clear that choice of computing model can have a significant impact on the viability of a given data management strategy. As these models evolve, we suggest that their feasibility in terms of the then-current RD45 strategy is reviewed. A series of white papers[18] are being produced by RD45 – of which this document is a member – which propose guidelines for various issues concerning the deployment of a data management system based upon Objectivity/DB. These guidelines are deliberately conservative and avoid the use of features that we believe are either immature, with which we have insufficient experience or where further testing is required. It is expected that these recommendations will evolve with time, but the adoption of proven technology is strongly recommended.

An example of such an issue would be the use of collaboration-wide Objectivity/DB federations in a computing model. As a federation is a tightly-coupled entity, any computing model based upon the deployment of a single federation[19] across multiple sites would

[17] This document formed the basis of the presentation made at the October 1998 RD45 workshop, which can be found at http://wwwinfo.cern.ch/asd/rd45/workshops/oct98/presentations/jamie/Risks/index.htm.
[18] See http://wwwinfo.cern.ch/asd/rd45/reports.htm.
[19] Until we have understood how such issues could be managed, our recommendation is to avoid federations distributed over more than a few, closely collaborating, sites.

necessarily require very close collaboration between all institutes involved on operating system, compiler and Objectivity/DB versions.

Work on LHC Computing Models should be closely coordinated with that undertaken by RD45. The risks associated with each model should be evaluated and the risk factor should be included as part of the overall assessment of the viability of the model.

Close links between the MONARC projects and RD45 are maintained. RD45 presentations at MONARC meetings can be viewed at http://www.cern.ch/MONARC/docs/rd45_presentations.html.

11.3 Choice of Technology

The solution currently proposed to the needs of the LHC experiments in terms of data management is based upon an ODBMS coupled to a Mass Storage System (MSS). In this document, we only consider the risks associated with the ODBMS layer, although clearly the risks associated with the MSS layer need to be evaluated. Alternative solutions to an ODBMS, such as the use of an RDBMS, an Object-Relational solution, "light-weight" or other persistency mechanisms, or conventional file-based I/O have a number of significant restrictions that made the choice of an ODBMS more appropriate. However, in terms of a fall-back solution, one of these alternatives might turn out to be viable. For example, Ardent Software has recently announced an ODMG-compliant C++ wrapper for a number of RDBMS products, including ORACLE, Sybase and Informix. There is a good chance that at least one of these products, or at least similar technology, will continue to exist in 10-20 years. Would such a wrapper offer the functionality that is required? Can the underlying technology offer sufficient performance and scalability to meet the needs of the LHC experiments? What are the chances that the wrapper or equivalent software will continue to exist, even if the underlying RDBMS is still available?

RD45 should evaluate the functionality, performance and scalability of ODMG-compliant C++ wrappers to ORACLE with a view to developing a potential fall-back solution.

Based on information obtained from ORACLE at a data management workshop at SLAC[20], it appears unlikely that the RDBMS vendors will attempt to address the PB region until there is sufficient commercial demand. This is not expected to occur in time for the 2001/2 choice of a solution for the LHC startup in 2005. In addition, the solutions that are currently available – typically C++ wrappers – are provided by small companies with no long-term guarantee of survival. Hence it is felt that no active work in this area is appropriate at this time, although the evolution should clearly be followed.

[20] See http://www-user.slac.stanford.edu/rmount/dm-workshop/main.htm.

It is clear that the database market will continue to evolve. Existing products will provide additional functionality and new ones will emerge. Other database paradigms may be developed, perhaps even more suitable to HEP than the ODBMS model.

RD45 should continue to monitor and evaluate developments in database products and alternative database solutions. This includes obtaining relevant industry reports, as well as making product evaluations.

This continues to be part of RD45's activities. To this end, RD45 intends to participate actively in the 1st European Workshop on Object Databases[21], whose theme is central to this discussion.

The ODMG standards for ODBMS products are implemented, at least partially, by a number of database vendors. The ODMG is aligning itself more closely with the OMG, and may well merge with the OMG, e.g. become a sub-group, in the future. At the same time, it is taking steps towards formal ISO approval of the standards that it has developed. It is therefore likely to remain at least the de-facto standard in this area, if not become the de-jure one.

RD45 should continue to monitor the activities of the ODMG and continue to push for vendor-compliance.

The ODMG appears to be diverging from the pure and simple definition of Object Database standards. Currently, their main activities focus on Java – C++ and Smalltalk work appearing to be of somewhat lower priority. In 1998, the group swapped "Data" for "Database", suggesting that they were addressing implementations other than ODBMSs, such as interfaces to RDBMSs. A new revision of the standard – ODMG V3 – is planned for later in 1999. However, increased vendor compliance – particularly as regards the C++ binding – is expected to proceed slowly.

With the partial exception of Versant, all currently known alternatives share a common draw-back with respect to Objectivity/DB, namely the lack of the federation concept. Conventional file-based approaches, be they based on sequential or random access, have the further disadvantage that they require at least one, if not two, additional layers. These layers provide a file (database) catalogue and event directory. Existing experience in HEP suggest that several man-years are required to develop and maintain each of these layers, whilst still not offering the functionality offered by a fully distributed federation.

An estimation of the manpower involved in developing and maintaining the additional layers required over file-based or multiple independent database approaches should be made. An evaluation of the loss of functionality with respect to a fully-distributed federation should be included.

[21] See http://www.disi.unige.it/conferences/oodbws99/.

No such estimate has yet been made directly by RD45. It is felt that experience at LEP, and that gained by experiments such as CDF and D0 – and others who have adopted similar strategies – will provide sufficient information on these approaches.

Of course, the best guarantee that a given technology will continue to exist is demand. This does not guarantee that a specific implementation will always be available, but that equivalent functionality will be supported. A key area where we currently have atypical requirements is that of scale: few applications require scalability into the PB region with correspondingly high data rates. On the other hand, storage requirements are rapidly increasing everywhere. Numerous industry analyses predict that databases of the order of 1PB will be relatively commonplace in the early years of the next millenium. Some predictions suggest that they will be as large as 10PB. Although the storage capacity that will be supported by commodity databases is hard to predict, it is almost certain that VLDBs will require distributed databases with relatively tight coupling, e.g. consistent schema, inter-database references. This is an area of particular importance to us, and one where we can predict a demand that goes far beyond the realm of HEP.

A revised version of the report "Object Databases and Mass Storage Systems: the Prognosis", CERN/LHCC 96-17, should be produced to help assess the likelihood that the requirements of the LHC experiments in the area of data management will be met.

A revised report is currently being prepared by the "PASTA" technology tracking team. The PASTA team covers essentially all of the storage-related issues discussed in the Prognosis, with the exception of ODBMSs. An update concerning ODBMSs is available in "Persistent Object Manager Choices" CERN/RD45/99/01 [4].

11.4 Choice of ODBMS Vendor

There are clearly risks involved with the choice of "supplier" of a given solution. This is true regardless of whether the solution is developed risks in-house or acquired commercially or otherwise, particularly given the extremely long time-scales of the LHC. It is impossible to absolutely guarantee the survival of a given vendor over some 15-20 years, just as it is impossible to guarantee the survival of a home-grown solution over such a long period of time. Risks involved with the choice of a vendor include not only its long term survival potential, but also its focus on a given market segment. In both of these cases, it is important to understand if the vendor considers the market segment to be of importance, and if it is of sufficient size as to guarantee at least the medium-term survival of at least one solution satisfying the requirements of the market segment.

Over the past few years, Objectivity has refocussed its strategy on the high-end, concentrating on areas where its product is architecturally most appropriate. These include markets where there is a demand for scalability and performance – probably the two issues of highest priority to HEP. It currently dominates the scientific and technical market, in areas ranging from High Energy Physics to Astrophysics, Geophysics, medical physics etc and has a strong presence in the telecommunications market. There is no doubt that CERN

has had a significant impact on this change of focus, and in Objectivity's success in the scientific market place. Their success in the telecommunications market should also be of interest to us: here there is a strong requirement for distribution, performance and reliability. There is also a significant amount of money: current estimates predict that the global satellite-based telecommunications market will be worth some $40 billion by the year 2010. Objectivity's involvement in the IRIDIUM[22] project and likely involvement in other such efforts is likely to provide them with significant revenue – a crucial component to ensuring long-term viability.

RD45 should develop a very clear understanding of Objectivity's marketing strategy and financial model. RD45 should continue to be pro-active in helping to define the future direction of Objectivity, by participating in Developers' Conferences and Technical Fora. Visits should be made to a number of key Objectivity customers to understand their strategies for risk management. Suggested customers include Motorola/Iridium, Nortel, Fisher-Rosemount and the various Astrophysics projects. The possibility of collaborating with other Objectivity customers, including an assessment of benefits, should be evaluated.

Objectivity believes strongly in the message preached in Geoffrey Moore's "Crossing the Chasm" book [6]. As such, they believe that it is necessary to concentrate on, and dominate, one particular market, which will then enable them to move on to new markets and thus expand. During 1998, there were consistently profitable and doubled the number of new customers with respect to the previous year. The evolution and stability of the company will continue to be closely monitored.

Given that it is impossible to guarantee the survival of any vendor over the timescales required for the LHC, a more pragmatic approach should be applied. We need to be aware of the issues involved in a potential migration to a different product, including the costs and timescales involved. Based on the experience gained for past migrations in HEP, and evaluations of alternative products, such as O_2, ObjectStore, POET and Versant, it is clear that such a migration require a significant effort and would have an impact on both Computing and Object models of an experiment. The current fall-back solution to Objectivity/DB, for technical reasons, is considered to be Versant. This product also supports a distributed database architecture, although is in a number of ways less suitable for our purposes than Objectivity/DB. However, at the time of writing, Versant's financial situation is somewhat unhealthy and cannot be considered a viable long-term alternative – there is no clear reason to believe that Versant has a better chance of survival than Objectivity. What is more relevant is whether at least one company that provides a viable solution will continue to survive and what are the implications of adopting an alternative solution.

RD45 should develop a clear understanding of the issues related to the use of an alternative ODBMS product, including issues relating to the lack of support for the federated database concept of Objectivity/DB.

[22] See http://www.iridium.com/.

It is felt that the concept of a federation is fundamental and important distinction between a "catalog of files/DBs" approach and an "ODBMS" approach. At a technical level, Versant remains the only realistic alternative ODBMS to Objectivity/DB. However, the company has suffered poor financial results recently, which unfortunately is probably as indicative of the ODBMS market as a whole as it is of Versant as a company.

The next 1-2 years are likely to be a critical period for the ODBMS market and Objectivity in particular. Can the company grow from its current size to have a true international presence? Can it survive the transition from private to public company? It is likely that a new CEO will be appointed on a similar timescale – how will this affect the company's stability? What if the company was to be acquired by another vendor, particularly one not interested in the current technology?

These are all questions that are hard to answer and clearly the situation needs to be followed closely. However, given the importance of the product to CERN, we should consider how we could maximise Objectivity's chances of survival and minimise our risks. As the past as shown, CERN's use of Objectivity/DB has had a positive effect on the company as a whole. We should therefore consider which activities are both in the interests of the organisation and could have a beneficial effect on Objectivity, such as participation in relevant conferences (IEEE Mass Storage Symposia, VLDB, OOPSLA etc.)

RD45 should continue to participate in relevant conferences and make presentations on our experience with ODBMS technology.

Presentations have been accepted at the 16th IEEE Mass Storage Symposium and the 1999 SIGMOD conference and have been submitted to VLDB '99. Participation is also planned at the 1st European Workshop on Object Databases.

It is clear that such steps cannot guarantee the survival of the company, and we should prepare for the worst-case scenario. As such, we should ensure that we have access to the source code and build procedures in case the company fails or is acquired.

CERN should add an ESCROW-like clause to the contract with Objectivity that guarantees access to the source code and build procedures. This procedure should be verified at the time of each major release of the product. It should be possible to invoke this clause even if the company continued to exist, but was unable to support the product in a suitable manner for the HEP community.

Attempts to agree on a revised contract with Objectivity were not conclusive – pending delivery of the outstanding enhancement requests, any discussion of additional licenses, and hence contractual changes, is on hold. It now appears optimal that any significant increase in the number of licenses be closely tied to the choice of production system, foreseen for around 2001. Nevertheless, a small number of licenses, e.g. for Linux or the HPSS interface, may still be required in the interim.

Even with access to the source code and build procedures, the issue of long-term support needs to be considered. Given that several HEP laboratories are using this technology, we should consider the possibility of sharing resources to provide medium-long term support, or of contracting the support out to a 3rd party.

RD45 should develop strategies whereby Objectivity/DB could be supported should the above clause need to be invoked. The cost of such support should be estimated, whether provided in-house, as a collaborative effort with other HEP laboratories and/or Objectivity customers, or via a 3rd party.

Without access to the source code, such an estimate is essentially impossible to make. Nevertheless, it is believed that the current support team in IT/ASD would be able to support and perhaps even develop Objectivity/DB if given access to the code base. It is our firm recommendation that any decision to use Objectivity/DB for the production period of the LHC be tied to a source code license or, at very least a "working-ESCROW". Such an agreement would ensure that CERN personnel were allowed to build each major release on e.g. Linux and NT to verify that the material held in ESCROW was usable and complete.

We have seen that a given release can be used for something like 1-2 years, before compiler and operating system changes require a new version. The above strategies should be sufficient to ensure that we are able to use the product for at least one complete change in operating system and compiler versions and hence provide time for a migration strategy to be developed and implemented. It is certainly not impossible that the product could be supported throughout the entire lifecycle of the LHC. Such a scenario would certainly be preferable to developing a product from scratch and almost certainly offer advantages over migrating to an alternative product. Parts of the product that are of little or no interest to CERN or the HEP community could be dropped, such as the SmallTalk binding.

An alternative option would be to initiate a joint project with Objectivity and other HEP laboratories. This could allow us early access to the source code and provide a framework in which enhancements of interest to HEP were implemented using HEP resources.

Another option suggested by Objectivity would be for CERN (HEP) to take a stake in Objectivity and thus have a vote on the Board.

Ways of increasing the chances of Objectivity's survival as a company and/or product should be investigated. Proposals should be made whereby we can be reasonably confident that the software will continue to be available for the lifetime of the LHC, even if supported in-house of via some other mechanism.

In 1999-2000, the Motorola IRIDIUM project goes live, and several hundred TB of data should be collected in Objectivity/DB in various HEP experiments worldwide. In addition, numerous presentations of CERN and RD45-related work are planned for conferences such as the IEEE Mass Storage Symposium, VLDB, SIGMOD and so

forth. Beyond these activities, there appears to be little that we can do to increase Objectivity's chances of survival.

However, probably the most realistic approach to ensuring long-term access to the Objectivity/DB software is to obtain a source code license. This would permit us to provide better support than using a binary license, offer greatly enhanced protection against failure or take-over of the company and provide a much better starting point for an eventual "HEP solution" than zero. The cost involved in such a license, which is likely to be significant, should be weighed against the extra protection and possible manpower savings in case of problems.

A source code license of Objectivity/DB should be investigated around the time of a decision to use Objectivity/DB for the production phase of the LHC, i.e. around 2001.

As described above, such an option continues to be strongly recommended by RD45.

It is likely that CERN will complete the port of Objectivity/DB to Linux and is already scheduled to help develop tests for a new test framework currently under construction. Hence, we are likely to gain much greater insight into the porting, build and test processes for Objectivity/DB, which would significantly increase our chances of providing medium-long term support in-house.

11.5 The Home-Grown Approach

Over the past 3-4 years, considerable experience in the use of an ODBMS and the issues related to large-scale object data management has been gained. A final fall-back alternative, should no suitable commercial solution exist in the long term, would be to design and implement a HEP-specific system, based upon the experience of the last few years and that we hope to gain from the production deployment of Objectivity/DB. The costs of developing and supporting such a system, particularly in the long term, are likely to be very significant. Given the importance of such a system to the research goals of the laboratory, it would be extremely important to first develop a clear set of requirements and to develop the system, should this be considered to be desirable, with reliability and "cost-of-ownership" as very high priorities. Such an approach could not address the needs of current experiments, but could perhaps be an alternative for a medium to long-term backup strategy.

An estimation of the manpower required to design, implement and support a HEP data management system capable of satisfying the key requirements of the Computing Models of the LHC experiments should be made. The possibility of joint collaboration with other laboratories should be considered.

Industry estimates of the effort required to develop a full ODBMS are typically of the order of 100-150 man-years. Such effort is clearly beyond the resources currently available in IT/ASD group. However, the GEANT-4 collaboration has clearly shown that world-wide collaboration between HEP institutes is possible, and that highly complex pieces of software can be developed in a fully distributed manner. In

addition, the considerable experience already gained in ODBMS technology would be a very important factor in both the design and implementation of a possible HEP data management system. Before any realistic estimate can be made, it is clearly necessary that a revised list of requirements is prepared. However, such a list is required in any case. It will be needed to help develop acceptance tests for an ODBMS (or equivalent product), to understand and better formulate our needs for further enhancements to Objectivity/DB and as part of the on-going work on LHC computing models. A first step in this direction could be the HEP ODBMS Testbed [7], discussed in another RD45 note.

Given the recommendations to establish an ESCROW agreement and eventually a source license, an estimate should also be made of the manpower needed to support and extend a solution based upon the Objectivity/DB code base.

11.6 Summary

Risk analysis and management is an on-going activity that is fundamental to the goals of RD45. The possible steps to evaluate and minimise the risks involved that are outlined in this report require a non-negligible investment, but are considered to be of highest priority.

We have listed above a number of areas where further investigations could be performed. We believe that, given the appropriate resources, a strategy can be developed whereby we can be assured that the required functionality remains available as long as is required and that such a strategy should be affordable both in terms of manpower and direct costs.

Given the approval of the LCB of the steps outlined in this document and allocation of the necessary resources, we would propose to implement the above recommendations and report back on the results to the LCB.

11.7 References

The RD45 reports listed below (e.g. CERN/RD45/xxxx) may be obtained via http://wwwinfo.cern.ch/asd/rd45/reports.htm.

[1] Assumptions concerning Distributed Data Management, CERN/RD45/9801.

[2] Guidelines for Setup of Production Objectivity/DB Federations, CERN/RD45/98/02.

[3] Guidelines for the use of Replication in Objectivity/DB, CERN/RD45/98/03.

[4] Wide-Area Objectivity/DB Federations, CERN/RD45/98/04.

[5] Persistent Object Manager Choices, CERN/RD45/99/01.

[6] Crossing the Chasm, Geoffrey Moore, Harper Business. ISBN 0887307175.

[7] A HEP ODBMS Testbed, CERN/RD45/99/06.