

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/LHCC 96-17
LCRB/RD45
8 February 1996

Object Databases and Mass Storage Systems: The Prognosis

The RD45 collaboration
CERN, Geneva, Switzerland

We present a statement of the Probable Capabilities of a HEP Persistent Object Management System based upon commercial Object Database Management Systems (ODBMS) and large-market Mass Storage Systems (MSS). Many of the capabilities mentioned in this report are expected to become available in commercial products long before 2004. However, it is felt unwise to attempt to extrapolate much beyond the year 2000, particularly when some people predict that much of our current computing infrastructure, namely PCs and workstations, will become obsolete before then. Thus, rather than extrapolate too far into the future, we prefer to make more realistic, and justifiable, predictions for a shorter time period, and update these predictions with time. This document has been produced in response to the revised milestone 1 set by the LCRB on the RD45 collaboration. The purpose of the document is to provide the appropriate technological input to the LHC Computing Model Working Groups.

1	Introduction.....	5
---	-------------------	---

2	Information and DBMS Trends, Beyond 2000	5
2.1	Introduction	5
2.2	See the Future in the Past	5
2.3	User-Level Abstraction.....	6
2.4	Transparent Storage	7
2.5	Proliferation of Computers and Communication	8
2.6	The Sum is Greater than the Parts	9
3	Disk predictions	10
4	Filesystem predictions	10
4.1	Filesystems for Unix.....	10
4.2	Filesystems for NT	11
4.2.1	A Peek at OFS	11
4.3	Parallel filesystems	12
4.4	Distributed filesystems	13
5	Tape predictions	13
6	Mass storage predictions.....	14
6.1	HPSS.....	15
6.1.1	HPSS Scalability Goals.....	15
6.2	ADSM features	16
6.3	Status of standards activities	16
6.4	Trends from the IEEE symposia on Mass Storage Systems.....	17
7	Networks	18
7.1	Tigers and Icebergs: Microsoft On-Line	18
7.2	The Race to Global Information Super-Skyway	19
7.2.1	Background	19
7.2.2	The System.....	20
7.2.3	The Market.....	20
8	Object Database Management Systems	20
8.1	The Object Database Manifesto	21
8.1.1	Additional features	21
8.1.2	Transactions	22
8.2	Federated Databases	23
8.3	Distributed Databases.....	23
8.3.1	Caching	23
8.3.2	Distribution	24
8.3.3	Replication	24
8.3.4	Import/Export facilities	25
8.4	Schema Evolution.....	25
8.5	Data Access	26
8.5.1	Naming Objects.....	26
8.5.2	Associations	26
8.5.3	Iterators	26
8.5.4	Indexing	27
9	Object Database Management System predictions.....	27

9.1	Status of the ODMG standard	27
9.2	Very Large Databases (VLDB).....	28
9.3	Integration of ODBMSes with Mass Storage Systems.....	29
9.3.1	Integration of Objectivity with Mass Storage Systems.....	30
10	Standards integration	30
11	LHC Computing Model scenarios	30
11.1	The Fully Centralised Solution.....	31
11.2	The Partially Distributed Solution.....	31
11.3	Distribution of Rawdata	32
11.4	Persistent Storage versus Recalculation of “DST” information.....	33
11.5	Access to complete events.....	33
11.6	Access to sub-events (sub-detector)	34
11.7	Access to individual components of an event (“Ntuple” style).....	34
12	Capabilities of a HEP Persistent Object Management System.....	34
12.1	The Reconstruction Process	35
12.2	Data Analysis.....	36
12.2.1	An Example Data Analysis Framework.....	37
12.3	Alternative data analysis frameworks	38
12.4	Other database applications	38
13	Conclusions.....	38
14	Appendix A - list of key HPSS features	39
14.1	Implementation Requirements.....	39
14.2	Design Standards	39
14.3	Scalability	39
14.4	General	39
14.5	Interfaces	39
14.5.1	Media Support.....	40
14.5.2	Robotics Support.....	40
14.5.3	File Management and Storage Management.....	40
14.6	System Management	40
14.7	Security.....	41
14.8	Other Features	41
15	Appendix B - list of priorities for ODMG V2.0	42
16	Glossary	43
17	References.....	44

1 Introduction

It is inevitable that the information presented in this report will become out of date. An examination of reports such as the CERN “green books”[1][2] and the MUSCLE report[3] shows that long-term predictions can be wildly inaccurate. Nevertheless, technology predictions are an essential component of any long term plan. In addition to predictions regarding the capabilities of a complete HEP persistent object management system, we provide predictions of the various components of the system, such as (parallel) filesystems, disk and tape technology, Object Database Management Systems and Mass Storage Systems.

The information presented in this document has been derived from a number of sources, including from company-confidential material. This document should therefore be treated as confidential. In particular, it should not be shown in its entirety to a vendor or third-party. A version suitable for distribution, but with correspondingly less information, will be made available through the RD45 web page, which can be accessed via the URL <http://wwwcn.cern.ch/pl/cernlib/rd45/index.html>.

For an introduction to Object Databases, the reader is referred to the books by Mary Loomis [10] and Rick Cattell [4], which should be read in order. Information concerning the Object Database Management Group can be found via the URL <http://www.odmg.org>. The Object Database Management Group standard for Object Databases is described in [16].

2 Information and DBMS Trends, Beyond 2000

[This section was written by Drew Wade, the founder of Objectivity and a co-founder of the Object Database Management Group. It is also available on the Web via the URL <http://www.objectivity.com/ftp/papers/2000.txt>.]

- See the future in the past
- User-level abstraction Transparent Storage
- Proliferation of Computers and Communications
- The Sum is Greater than the Parts

2.1 Introduction

Information management has always been a principal application of computers and will play an even stronger role in the future. It will support more complex information at levels of abstraction defined by users, provide distributed views over world-wide networks, and become integrated with communication systems. The result will be a synergy that makes electronic information management a part of every day life, integrated into many activities.

2.2 See the Future in the Past

How can we predict what role computer databases and information management will play in the coming decades? Lacking a time machine to visit the future and observe directly, all that remains is to

examine the past. This we can do in two ways. First, we can examine leading activities in research and attempt to predict how they will play out. Second, we can examine past trends and extrapolate them into the future.

While computers have popularly been thought of as fancy calculators, and while they do in fact serve many computational or number-crunching purposes, their widest use has been in storing, accessing, managing, and analyzing information. We examine three trends in computing and information management, and in each case extrapolate them to create a vision of the future that shows even greater involvement in computers to manage the information we live with.

2.3 User-Level Abstraction

The history of computing has been a steady advance from the language and communication level of the machine towards that of the user. The earliest machines required users to drop themselves to the level of the machine, to translate what they wanted done into the machine's language of 0's and 1's and the machine's operations of store, increment, etc. Next, high level languages allowed users to write mathematical statements describing operations, which were automatically translated into machine code. However, the user, or programmer, still had to translate his problem into this computer language. Fourth generation languages (4GLs) allowed end users who were non-programmers to access information without explicitly defining each step, but only provided certain built-in types of operations on pre-defined data types (tables).

The latest advance is object technology, which allows users to package together code and data into units, called objects, at higher and higher levels of abstraction. The user of an object need not know anything about how it's implemented to invoke the object's operations, and those operations can correspond directly to the user's desired application. By combining objects into composite objects, users can build more and more complex systems, without limit, and each user can work at his own level of abstraction.

The evolution of information management parallels this progression, starting from low level programs that directly read and write core memory, tape, and disk tracks and cylinders, followed by file systems that gave higher level views of byte arrays, records, and indices for improved performance. Databases started with this same model of records and indices, adding capability to recover from failures, to manage data integrity, and to control concurrent access by multiple users. These early database models (network, Codasyl) were at the level of the machine, much like the early programming languages. Later databases (relational) evolved a mechanism to perform ad hoc queries on the data, allowing the user to specify what he wanted, while the DBMS automatically figured out an optimized execution path based on dynamic indices, and returned the result. Although this added a higher level interface, just like the 4GLs, it did so at the cost of limiting the data to simple, flat tabular format and a few pre-defined operations. It was up to the user to figure out how to map his application operations and data structures, to flatten them into tables. The most recent trend in databases is to support objects directly, allow users to define, create, modify, and share objects with the traditional database facilities for recovery, concurrency, integrity, query, etc.

Although just begun, the object trend is now catching on rapidly in all aspects of computing, from graphical user interfaces (GUIs, icons such as Macintosh and Windows), to languages (C++, Smalltalk), analysis and design tools, operating system frameworks, and databases, too. Not only can users

build arbitrarily high levels of abstraction, but they can continue these same levels of abstraction from analysis through programming and into the database. By avoiding the need to translate from the application domain to programming constructs, users save time in telling the computer what they want (programming) and in executing the result (runtime). By re-using the same objects throughout and working directly at the application level of abstraction, they reduce errors.

As tools mature and users begin to appreciate their value, more and more libraries of off-the-shelf objects will become available. Today programming requires learning complex techniques for translating application problems to computer languages. Tomorrow it will simply be a matter of choosing pre-existing objects that can be combined to achieve the desired result, sometimes extending those objects. Today we are in transition, unlearning all that we have had to know to translate to files and records and tables and queues and operations such as join and project, while tomorrow's users will skip learning all that, and work directly in the domain of their application. There will continue to be professional programmers to build the core objects, but more and more non-technical end users will customize their own systems by dragging and dropping and linking pre-built objects.

These objects will be accessible in libraries of standard objects, widely available. Information, too, will be accessed the same way, via intelligent agents that can search for and manage the specific type of information that the user wants. Massive volumes of information will no longer be daunting, due to the ability of these higher levels of abstraction to manage the information for the users. Today we are just beginning to see the accessibility of world-wide massive information stores via the internet (e.g., all university libraries), but the result is overwhelming due to the lack of organization. With these intelligent objects, it will be manageable and usable for daily tasks.

The result will be many more direct users of computers, in a wide variety of professional, business, and home use, far more powerful uses of computers, far more reliable uses (because of re-using standard, tested, working components), and access to far wider varieties and amounts of information.

2.4 Transparent Storage

In the old view, as Nikolaus Wirth's famous textbook puts it, programming = algorithms plus data structures. Even more pointedly, programmers and computer users view programs and data storage mechanisms, including DBMSs, as distinct. Typically, different groups develop the two, manage and maintain the two. However, that division is in the process of going away.

This is somewhat analogous to the move to bitmap graphics systems from serial or parallel devices. With the latter, users put text and images on screen by writing special commands to the device, using a special library built by the device specialists. With the newer bitmap graphics devices, programmers view the screen as part of memory, just like any other data they address, and they can simply write to that memory, while somehow, the graphics system gets the results to the screen. This has resulted not only in richer and faster graphics, but in a complete change or paradigm shift in user interfaces, from text-driven to window-driven with menus, icons, mice, images, drag&drop, etc.

The analogy with information management works much the same. In the old way, users write a program, and when they want information, they make special calls to the I/O system, and when they want to save some information they make special calls again to put it back. This "get/put interface" usually also includes a translation of data format from the programmer's language (COBOL or FORTRAN)

to the DBMS format (ISAM records, tuples/tables). This translation is complicated, slow, and error prone. With the newer way, users view their information as part of objects and simply invoke operations on the objects, while the ODBMS transparently maintains the object persistently on disk, ensuring recoverability, concurrency management, and other facilities. There is no more dichotomy between the program and the data storage; there are just objects.

Even the structure of development groups, project management, is changing. Historically, it was common to have separate groups for creating the programs or algorithms and for creating the data structure (schema) and read/write code. With the new approach, that is not only unnecessary, but even counter-productive. Software system architecture tends to reflect the architecture of the organization that built it, so such an organization tends to create a separate layer isolating the programmers from the data access, which only slows access and limits functionality and concurrency. With the direct model, programmers and users just see objects and use them directly.

The result is that all programmers and all users of computers become users of DBMSs, because they're all directly using objects that are managed in secondary (and possibly tertiary, if that continues to exist...) storage, even though they never think about that storage or do much to explicitly control it.

Further, much processing that has traditionally been part of the application program is becoming part of the objects themselves, which are within the DBMS. Semantics, or meaning, including interpretation, rules of integrity, behaviour, all become directly part of the object. Where users of traditional databases and files had to think about what they wanted and how to get it and how to piece it together into application entities and what to do with it, users now and into the future, instead, simply invoke operations on objects directly. These operations themselves get the objects, perform standard processing, maintain integrity, etc.

This is analogous to the evolution that file systems provided. Before that, users thought about and mapped their application structures into physical units on the disk such as tracks, sectors, and cylinders. File systems provided a higher level of abstraction, so that users could think of records or arrays of bytes, which are automatically mapped to tracks, etc. Very quickly, users completely stopped thinking about tracks, etc. (except for a very small group of device driver programmers).

The result is that programmers and users no longer think of storage at all, but rather think of their application and the objects they use.

2.5 Proliferation of Computers and Communication

There has been an exponential advance in price/performance ratio of computers to the point where computers are not just fancy business or scientific machines, but are indeed found in televisions, refrigerators, and watches. This is likely to lead to a proliferation of computers to the point where they are everywhere and effectively freely available.

A corresponding advance has been occurring in storage media. Where core memory a couple decades ago required a refrigerator-sized, expensive box for 20K, today's PCs have perhaps 10MB ram and 1 GB on disk. We are storing larger and larger quantities of information on smaller and smaller devices

with faster and faster access (though seek time doesn't scale as fast as the others...) and cheaper and cheaper.

At the same time, technology for communications has been advancing at rapid rates, allowing more information to be moved from one location to another, faster and cheaper.

The combination points to a world of omnipresent computers, huge amounts of information located in all convenient points, with free movement between them. In fact, as Jim Gray once put it, it may well make sense to constantly stream information around. The internet already allows access to overwhelming amounts of information. This will only grow. And access to it will be faster. And massive amounts will be stored on large numbers of different sites, automatically replicated, accessed freely over high-speed communication, cached locally, shared with others.

In the old mode, information access was rare and difficult. Users hoarded information, required their own copies, guarded them locally. In the new mode, information access will be freely available. Although large caches will be kept locally, the massive amounts available and high speed communication will lead users to change to a mode of directly accessing remote data, transparently replicating and caching what they use most frequently today, though that local cache may change tomorrow, with constant refresh and update.

Already today, it is difficult for any business or profession to remain competitive without on-line, networked computer information access. Tomorrow, it will be minute-by-minute and second-by-second access to world-wide information, constant updates on market information, on business conditions, on data collected by others, updates and analyses instantly seen, and instantly acted upon.

2.6 The Sum is Greater than the Parts

What do these combined trends mean? What does this collective picture show? The everyday world will be as much changed by this growth in information management as it was by the industrial revolution. That revolution changed the world from manual labour, agriculture, and craftsmanship, to one in which most of us are involved in manufacturing and using manufactured goods. This revolution will change the world to one in which most of us are involved in processing information, and information will become the basic stuff of our daily lives, work and play.

From the future looking back, it will seem strange that only a few could use computers, because everyone will be using them in almost all aspects of work and play. It will seem strange that it used to take a cadre of specialist months or longer to get the computers to do what they wanted, because everyone will be doing it themselves, directly, instantly. It will seem strange than users spent time trying to find what they wanted, hiring services to locate desired information, shopping, travelling, mailing, waiting, because information will be at everyone's finger tips, updated instantly, accessed directly. It will seem strange that users spent time copying data, figuring out where to store their own copies, because now users simply process information as they like, without regard to where it is or where it goes, because it's dynamically and transparently available, and automatically replicated and reconfigured as needed for use in the office or on the beach.

Information will be the medium of business, of professions, even of entertainment, with intelligent agents automatically tracking changes users are interested in, sorting and caching locally desired information, and sharing changes and desired changes with others.

3 Disk predictions

CERN currently has several TB of disk installed on the CORE systems. This is primarily composed of 2 GB disks, although 9 GB disks are now being installed. Technology predictions from IBM¹ suggest that 20 GB 3.5" drives will be in production as from 1997, and 90 GB as from 2000. By replacing the existing CORE disks by 90 GB drives, one could provide some 250 TB of disk space. By delaying the purchase until 2003, when a newer generation of disk technology could be expected, 1 PB of disk space seems achievable.

Although it is not expected that the current 3.5"-based technology will continue - smaller form factors are expected to takeover, largely driven by the (portable) PC market, it is important to understand the commercial pressures on disk capacities and prices. Essentially every living person on the planet is a potential customer for disks, which translates to a vast market with correspondingly huge pressures. Thus, it is by no means impossible that the trend will be super-linear², and that disk farms well in excess of 1 PB will be possible by 2004 or shortly thereafter. By comparison, 1.2 GB of RA81 disk space (3 drives) cost more than 100 KSF (CERN price!) in 1985 - the original VXCRNA 8600 was delivered with only 6 drives! 4 years later, the disk capacity on the central VAXcluster had grown to 100 GB.

Thus, one can safely predict that disk farms in the multi-hundreds of TB, if not PB, range will be possible at the start-up of LHC.

4 Filesystem predictions

Below we present predictions for filesystem enhancements for Unix and NT systems. These are presented separately as the strategies are somewhat different - filesystems on Unix systems will continue to use files as the basic container, whereas on NT, the plan is to move rapidly to a database approach, whilst at the same time, removing existing limitations such as file and filesystem size.

4.1 Filesystems for Unix

64-bit filesystems will shortly become ubiquitous. These are already appearing on Unix systems - even on operating systems which are still 32-bit - and are expected to become prevalent within 3 years. With 64-bit filesystems, many of today's limitations will disappear. These limitations include the maximum file size (currently 2GB in a 32-bit filesystem) as well as the maximum filesystem size

1. Obtained from IBM's Storage System Division in San Jose - similar predictions have also been obtained from DEC.

2. In fact, IBM claim that the introduction of the magneto-resistive head (see <http://eagle.almaden.ibm.com/storage/oem/tech/eraheads.htm>), which they pioneered in 1991, resulted in a change of slope of the maximum areal density versus year of general availability curve.

(also 2GB). True 64-bit filesystems would theoretically permit all LHC data to be stored in a single filesystem, although this will not happen in practice due to other considerations.

Examples of 64-bit filesystems include XFS³ from Silicon Graphics. XFS provides a full 64-bit filesystem capable of scaling easily to handle extremely large files and filesystems that can grow to 1 terabyte (up to 9 million terabytes⁴ in successor releases).

XFS's major features include:

- full 64-bit file capabilities
- large files
- large filesystems
- large numbers of files
- sparse files (files with "holes")
- rapid and reliable recovery of filesystem structure using journalling technology
- integrated, full-function volume manager called XLV
- extremely high I/O performance that scales well on multiprocessing systems
- guaranteed rate I/O for multimedia and data acquisition uses
- back up of filesystems while still in use, significantly reducing administrative overhead
- compatibility with existing applications, EFS filesystems, and NFS

In summary, it is fair to say that not only do 64-bit filesystems with individual files of hundreds of GB exist today but also that 64-bit filesystems with individual files in the multi-TB range will exist long before LHC.

4.2 Filesystems for NT

A 64-bit version of the NT operating system is not expected until around the year 2000, primarily due to lack of market pressure. Nevertheless, Microsoft plan a number of significant enhancements to their current filesystems. The following article, extracted from the cover story of the August 1995 edition of Byte magazine⁵, explains Microsoft's strategy in some detail.

4.2.1 A Peek at OFS

There is a crisis in disks today. The FAT (file allocation table), which is the file system used on all DOS PCs, doesn't use all the space on them. Because the FAT has a fixed number of sectors, it must increase their size as disks get larger.

Cairo will introduce an improved file system that can address logical mass-storage devices as large as 408 million TB. That's about 418 billion GB or 427 trillion MB.

But the real challenge for future file systems is to make it easier for users to manage their mass storage. Already, people are losing track of files on 200-MB hard drives. It's obvious that the current

3. See <http://www.sgi.com/Technology/xfp-whitepaper.html>.

4. Or 9 exabytes (9×10^{18} bytes) - more than enough for all LHC event data!

5. Available via the URL <http://www.byte.com/art/9508/sec6/art3.htm>.

model of organizing files into more deeply nested thickets of directories and subdirectories simply won't keep up.

Files aren't just files anymore. With compound documents, files may contain embedded objects in diverse formats that are linked to the applications that created them. For example, a Microsoft Word document might contain tables linked to Excel and pictures linked to Paintbrush. If you want to view all the Excel tables you created over the past six months, today's file systems are inadequate because the smallest items they catalog and retrieve are files, not objects. Users need a finer-grained file system that lets them find, retrieve, and manipulate objects independently of the file structure.

Microsoft's answers to these problems are OLE structured storage and OFS (Object File System), an extension to NTFS (NT File System). OLE structured storage is like a file system within a file. The compound file is subdivided into a tree structure of subfiles called stream objects and sub-subdirectories called storage objects. Your data is stored in the streams; storage objects may contain streams or other storage objects.

This might appear to complicate matters even further, because it adds yet another tree to the file hierarchy. That's where OFS comes in. Microsoft recently announced OLE DB, an interface layer that connects OLE to back-end databases. That database can be a flat-file database, a relational database, or a file system organized like a database. In effect, OFS will be a relational database that can be searched and sorted using tried-and-true query methods.

OFS is extensible, so you (or a program) could add new fields. Today's DOS FAT has only a few fields for such attributes as time/date stamps, file size, and read/write flags. But OFS could have fields that store the name of the person who created the file or object, the formats of the objects, or just about anything else that would help you manage your information.

Some elements of this technology are found in Unix, OpenVMS, System 7.5, the Newton OS, and OpenDoc's Bento-format compound files. Indeed, even some current Windows applications (e.g., Word) let you tag a file with additional attributes. But for most PC users, OFS will be the biggest leap forward in file management since the introduction of subdirectories in DOS 2.0. And the implications for corporate data systems are even more remarkable by enabling the tracking and searching of data in compound documents.

4.3 Parallel filesystems

In addition to the move to 64-bit filesystems, we see a trend towards parallel filesystems. Parallel filesystems have been used in production at CERN on the CS-2 for all bulk-data access since the beginning of 1995, including use by NA45, NA48 and NA49, and are expected to be in production on the SP-2 by the end of 1996. The principle advantage of parallel filesystems is I/O performance. Single stream throughput of around 100 MB/second can be expected by the end of 1997⁶. Single stream throughput of several hundred MB/second can be expected by the year 2000. Aggregate throughput in the multi-GB/second range can also be expected.

6. In fact, tests with the parallel filesystem on the CERN CS-2 machine have already demonstrated throughputs of 60 MB/second!

For more information on parallel filesystems, see for example the web page of the Portable Parallel Filesystem project, accessible via the URL <http://www-pablo.cs.uiuc.edu/Projects/PPFS/ppfs.html>.

4.4 Distributed filesystems

Distributed filesystems⁷, such as NFS and AFS, are already in wide-scale production use today. The main improvements that one can expect to see over the coming years include

- integration with mass storage systems, via a migrating filesystem interface
- improved caching algorithms

On the Unix side, the main change that is expected over the next few years is a gradual migration to DCE (DFS)⁸. However, it is widely predicted that the community will move away from Unix towards Windows/NT or its successors, where much more significant changes are likely to occur.

5 Tape predictions

The commercial pressures on tape products are by no means as strong as for disks and are primarily driven by backup and data archiving requirements. The number of sites with data archival requirements that cannot be met by the latest generation of tape products is relatively small - far too small to finance the development of a new generation of tape products. Although data backup requirements grow with installed disk capacity, the move towards distributed computing and I/O scalability issues tends to favour the development of "low-end" tape products, such as the Digital Linear Tape or DLT, originally developed by DEC and now manufactured by Quantum⁹, and not high reliability/capacity/throughput devices required in the scientific market-place. Thus, although we can expect several improvements over the latest generation of tape products, the changes will be less dramatic than for secondary storage.

The latest generation of tape cartridges from IBM, the Magstar 3590 (NTP)¹⁰, has a capacity of 10 GB per volume. This will increase relatively slowly, to more than 30 GB per volume by 2000. Capacities of 50-100 GB per volume by 2004 are possible, provided the necessary market pressures exist to drive the required development.

STK's Redwood¹¹ cartridge, which uses D-3 helical scan technology, imported from the video industry, rather than longitudinal recording as favoured by IBM, currently has a capacity of 50 GB per volume uncompressed. Cartridges with 100 GB capacity are expected by 1997, and 200 GB or more by

7. It is not clear if distributed filesystems actually offer any advantage in a distributed database environment. In fact, it is possible that the caching mechanisms of the filesystems may even interfere with those of the database! However, this area is a topic of much research work, both at the level of more intelligent, configurable, caching algorithms at the level of the filesystem [21], and in terms of user-configurable operating systems [26].

8. DFS is already supported by the main vendors of interest to CERN, namely DEC, HP, IBM, SUN and SGI. For more information concerning DFS, including performance information, see <http://www.cray.com/PUBLIC/product-info/sw/dce/perf.html>.

9. See http://www.quantum.com/products/product_guide/dlt2500.html for more information.

10. See <http://www1.ibm.com/HTML/SPEC/goem7065.html> for more information.

11. See <http://www.stortek.com/StorageTek/redwood1.html> for more information.

2000. By 2004, cartridges with a capacity of 500 GB (magnetic) or 1 TB (optical) with throughputs of 50 MB/second and 100 MB/second respectively may be expected.

Predictions for optical tape, given at the 14th IEEE symposium on Mass Storage Systems, foresee 100 GB per volume with transfer rates of 12.5 MB/second in 1997, 300 GB in 1998 and 1-5 TB in 1999 with perhaps 20-30 MB/second throughput.

Other future technologies include a 668 GB cassette being developed by NHK in Japan, with a phenomenal data transfer rate of 1.5 Gbit/second, prototypes of which are supposed to appear in 1997! Thomson-CSF are currently developing a very high-capacity/throughput tape device, primarily for the consumer market. The idea is that the device, which uses an interesting combination of a magnetic matrix head for writing, and an optical head for reading, both built out of commodity components, will be built into televisions and used to down-load films during off-peak hours, which can then be played back upon demand. A new, recordable, video-CD device, compatible with existing audio CDs and slated to obsolete existing tape-based video technology much as CDs did vinyl records, is expected to appear in consumer quantities towards the end of 1996. If such a device were ever adapted to the computer market, then it could offer an interesting alternative to magnetic or optical tape products.

Tape robots with a capacity in excess of 1 PB can be constructed today, e.g. using the STK silo, which has a capacity of 6000 cartridges per silo and can be configured in a 16 by 16 silo arrangement, or the Sony "PetaSite" tape library, based on ID-1 recording technology - the same tape technology as used by the NA49 collaboration. The expected increases in cartridge capacity should help to make such robots more affordable, although the important questions of data access and throughput should not be ignored - this is another area where parallelism will almost certainly be required.

6 Mass storage predictions

The Mass Storage market is changing considerably. Many of today's sites with a requirement for Mass Storage Systems will be able to satisfy this requirement with secondary storage by the end of the decade. The current "rule of thumb" as to when a Mass Storage System becomes a requirement is in the 1-5 TB range. This will increase to 100 TB or more by the year 2000. It is important to point out that mass storage only becomes a requirement when it is no longer feasible, for technical or financial reasons, to store data on secondary storage and is not a requirement in its own right.

This does not mean to say that the requirement for Mass Storage Systems will disappear. As it becomes technically feasible to store ever increasing volumes of data, new applications, which today are not viable, will become possible. Some such applications, such as constant monitoring of the visible universe using phased-array telescopes, can generate Petabytes of data per day!

Unfortunately, the commercial mass storage market is not healthy¹². Arguably only one product has ever been a "commercial" success, namely the Common File System (CFS) developed at Los Alamos National Laboratory in the US, and marketed by General Atomics as Datatree¹³. Other Mass Storage

12. Indeed, the development of at least one commercial mass storage system has been cancelled due to lack of sufficient market.

Systems¹⁴ include Unitree, also originally marketed by General Atomics, E-MASS from E-Systems, AMASS, now also from E-Systems, the Open Storage Manager, originally from Interactive Systems (later known as Lachman Technology and subsequently taken over by Legent and then Computer Associates), Large Storage Configurations from LSC Incorporated, IBM's ADSM product and finally the High Performance Storage System (HPSS).¹⁵

Only two of these systems, namely ADSM and HPSS, claim scalability to the region required by LHC - the multi-PB region. In our opinion, none of the other systems mentioned above are worth considering as candidates for building blocks of an LHC-era persistent object management system.

In order to gain some insight into the possible capabilities of a future Mass Storage System, we list below the features of HPSS, and of the current version of ADSM.

6.1 HPSS

The High Performance Storage System project (HPSS) is a collaborative development effort between a number of industry, government and research sites. The objective of the HPSS project is to develop the next generation, standards based, distributable, hierarchical storage system focused on scalability, in size and performance, and modularity of software components.

The list of key HPSS features is summarized in Appendix A (see section 14 on page 38). The HPSS web page can be accessed via the URL http://www.llnl.gov/liv_comp/nsl/hpss/hpss.html.

From our point of view, the most important features of HPSS are related to scalability, and these are reproduced below. HPSS is the only Mass Storage System that is attempting to address the very high end. It is our conclusion that a commercial offering in this domain is extremely unlikely - there is simply insufficient demand to justify the investment required to produce such a product - indeed, this is one of the reasons for the existence of the project.

6.1.1 HPSS Scalability Goals

- Storage capacity to multiple petabytes
- Throughput per transaction of multiple gigabytes per second
- Bitfile sizes up to 2^{64}
- Name space entries
- Number of simultaneous users
- Introduction of additional devices
- Software striping
- Distribution and multiprocessing of servers

13. CFS/Datatree was deployed at some 20 sites, but it is not clear if any of these ever paid a license fee!

14. See <http://www.cern.ch/~hepmss> for information on mass storage systems, including pointers to known deployments in HEP and related fields.

15. For more information on known mass storage systems and deployments, see the HEPMSS World-Wide-Web page, accessible via <http://www.cern.ch/~hepmss>

6.2 ADSM features

IBM's ADSTAR Distributed Storage Manager (ADSM)¹⁶ provides backup/restore, migrate/recall and archive/retrieve clients for numerous platforms, including Unix, PCs and mainframes. All of these clients use the services of the same Mass Storage System, which is based upon version 4 of the IEEE Reference Model for Mass Storage Systems. Database technology is used throughout the system, which is designed to handle up to a petabyte of storage in a distributed, heterogeneous environment.

ADSM has won a number of contracts recently for large-scale mass storage systems. The European Centre for Medium Range Weather Forecasting (ECMWF), for example, has recently acquired an IBM-based hardware/software solution that should cater with some 200 TB of data by the year 1998.

As IBM is a development partner in the HPSS consortium, one might expect that the major features of HPSS will eventually appear in a release of ADSM¹⁷. The significant difference between the two products is the market that they are trying to address. ADSM is aimed at the large scale workstation and PC backup market¹⁸, whereas HPSS is aimed at the specific needs of "lunatic-fringe" sites, which have exceptional mass storage requirements, both in volume and throughput.

It is indeed a concern that IBM might decide to abandon the high-end market altogether with its ADSM product, and hence involvement in the HPSS consortium would appear a sensible, if not mandatory, insurance policy.

6.3 Status of standards activities

There are a number of standards activities related to mass storage. These include relatively low-level standards groups, such as the Posix Removable Media Group, as well as higher-level activities such as the Data Management Interfaces Group (DMIG) and the IEEE Computer Society Storage System Standards Working Group (SSSWG). DMIG has made reasonable progress towards a standard API - the DMAPI. The goal of the DMAPI is to permit data management applications, such as migrating filesystems, to be built without making changes to the kernel or filesystem code. So far, the API has been incorporated into products from SGI, Convex, HP, Veritas, EMASS, Hitachi, Auspex and Epoch.

The situation regarding IEEE SSSWG is somewhat less positive, although things have looked up in recent months. After many years of ground work on the model, the working groups turned their attention towards individual standards, based upon the various components of the reference model. It was initially intended that drafts of these standards would become available by the middle of 1995. At the time of the 14th IEEE symposium on Mass Storage Systems, it looked as though the SSSWG effort would fold completely. Indeed, since that date, a number of key players have left the standardization effort. Nevertheless, the group continues to meet, and the fact that the draft Physical Volume Library

16. The ADSM home page can be accessed via the URL <http://www.storage.ibm.com/storage/software/adsm/adsm-home.htm>.

17. Indeed, such statements have been made by IBM to ECMWF (private communication).

18. It remains to be seen whether a system such as ADSM is truly scalable to the multi-PB region, or whether one can use multiple instances of a low/medium end solution to handle the high-end.

(PVL: cf. TMS) specification appeared late in 1995¹⁹. Since that time, a draft Physical Volume Repository (cf tape robot) specification has also been produced²⁰. These recent developments are cause for guarded optimism.

Fortunately, the situation is not quite as grim as one might deduce from the above. The SSSWG was mainly concentrating on internal interfaces, whereas the user community is primarily concerned with exposed interfaces. De-facto standards exist for device support, based on industry acceptance of the STK storage silo and the associated tape management software, and IBM tape devices. For example, not only do IBM's tape robots support the STK interface, but STK's new Redwood device has been certified (by IBM!) for use with their ADSM product via its 3490 emulation. One can safely conclude that no large market devices are likely to emerge over the next few years that do not conform to these de-facto standards.

The other exposed interface of concern is the API. Although no standard exists, the draft X/Open Backup Services Application Program Interface (XBSA)²¹ would appear to provide the necessary functionality. This draft standard is largely based upon the IBM ADSM API, and is very close to that offered by the Computer Associates OSM product. The X/Open XBSA will be supported in a future release of ADSM, and probably also in HPSS.

6.4 Trends from the IEEE symposia on Mass Storage Systems

The IEEE Computer Society has been holding symposia on mass storage systems for nearly two decades. An analysis of the themes of these symposia over the last few years reveals a clear trend:

- Crisis in Mass Storage (1990) [18]
- Distributed Storage Environments (1991) [20]
- Putting all that Data to Work (1993) [21]
- Towards Distributed Storage and Data Management Systems (1994) [22]
- Storage - At the Forefront of Information Infrastructures (1995) [23]

The proposed theme for the next symposium, scheduled for early 1997, is the integration of databases with mass storage systems.

From these themes, we can not only see a clear move towards distributed systems, now taken as read, but also a move from hardware-oriented concerns towards higher-level data management issues. A large fraction of the papers presented at the 14th symposium^{22 23} mentioned the use of databases, and many stressed the importance of data management. Major new projects, such as NASA's Earth

19. The PVL draft is available via the URL <http://mitrews.gsfc.nasa.gov/pvlapi/contents.html>.

20. The PVR specification can be accessed via http://www.arl.mil/IEEE/pvr_api.html.

21. See <http://www.xopen.org/public/pubs/catalog/p424.htm>.

22. The proceedings of the 14th IEEE symposium on Mass Storage Systems can be accessed via the URL <http://www.computer.org/conferen/mss95/mss14.htm>.

23. Surprisingly, the majority of the presentations made at the IEEE symposia are from sites with mass storage requirements smaller than the installed disk capacity on the Shift facility at CERN! For example, two of the NASA sites, with large, man-power intensive, mass storage systems, have a total installed capacity of 1.2-1.3 TB! This is not to say that more demanding requirements do not exist. However, these are often met by bespoke systems, as has typically happened in the past for military systems.

Observing System Data and Information System (EOSDIS - "Mission to Planet Earth"), are based on a combination of database (not always object-oriented) and mass storage technology. We expect to see these trends continue - future data management systems will be based upon database technology whereas hardware components will become "plug-and-play" commodity items. In this context, we mention also Microsoft's plans, described above (see section 4.2 on page 10), for a future filesystem which incorporates database technology.

7 Networks

Although many people predict that networking bandwidth will not increase significantly, or perhaps more accurately that any increase in bandwidth²⁴ will be accompanied by a similar - or even greater - increase in demand, such that network saturation will continue, there is a school of thought that predicts that networks will soon become ubiquitous, and that consequently the commercial pressures will be sufficient to guarantee that adequate bandwidth is made available. Examples of global network projects include the Microsoft Interactive Television project and the Motorola Iridium project, both of which are described briefly below²⁵.

7.1 Tigers and Icebergs: Microsoft On-Line

Today desktops, tomorrow the world. That sums up Microsoft's global OS strategy. Microsoft is preparing for a future where computing devices of all types are ubiquitous, networked, and part of our daily routine.

For a while, it looked as if PDAs (personal digital assistants) were the next big thing. But judging from Apple's struggles with the Newton, it will be a while before PDA technology catches up to expectations. That's a fortunate reprieve for Microsoft, which is having trouble scaling Windows to fit on today's palmtops.

The next battlefield could be TV set-top boxes for interactive broadband networks. Someone has to provide the software for the head-end video servers, the network switching equipment, and the millions of TV set-top boxes. Why not Microsoft?

Microsoft's data superhighway project runs under two code names: Tiger and Iceberg. Tiger, now dubbed MMS (Microsoft Media Server), is the video server that can spool independent streams of TV and video programming to thousands of home and business subscribers. Iceberg is the distributed OS that will run Tiger. Both technologies are undergoing small-scale trials.

The official name for Iceberg is MITV (Microsoft Interactive Television). Essentially, it's a distributed OS for the world's widest WAN. The foundations of MITV are familiar: Windows NT, Win32,

24. In fact, Telecoms companies and others, such as electricity companies, cable TV suppliers, rail-track owners etc. are laying fibre as fast as possible.

25. Other exciting new developments include the IEEE 1394 Home LAN, supported by Intel, MicroSoft, Sony etc., with speeds ranging from 100 Mb/second up to 1 Gb/second.

OLE, and COM (Component Object Model). But like a real iceberg, the bulk of this mass is submerged. A simple user interface keeps people from realizing they're channel-surfing with Windows. New security features in MITV protect the integrity of back-channel communications, so users can pay bills, manipulate bank accounts, and order merchandise right off the screen.

Published APIs will let third-party developers write applications for the set-top boxes, just as they do now for PCs. They'll use familiar tools: enhanced versions of Microsoft's Visual Basic and Visual C++. Most of the OS and all the applications software will be automatically downloaded over the network into the set-top box when users switch on their TVs. Only a small amount of boot code will reside in the box's ROM; this reduces costs and makes field upgrades transparent.

A related piece of this strategy is the Microsoft Network, the new on-line service that's integrated with Windows 95. It, too, will be a pathway for remote banking, home shopping, and content delivery. The two main differences are that the Microsoft Network is targeted at PCs instead of TVs, and it's designed to work over the relatively low-bandwidth network of the telephone system.

Because mouse potatoes are more open to new technology, the Microsoft Network will be a good test market for new services that may later be offered to couch potatoes. It guarantees that no matter which pathway into the home emerges as the most important--PCs or TVs--Microsoft will have all bases covered.

7.2 The Race to Global Information Super-Skyway

7.2.1 Background

The IRIDIUM²⁶ system is a satellite-based, wireless personal communication network designed to permit any type of telephone transmission - voice, data, fax, paging - to reach its destination anywhere on earth, at any time. It will revolutionize worldwide communications in the commercial, rural and mobile sectors by providing portable universal service. IRIDIUM is expected to become operational in December 1998.

Subscribers will use pocket-size hand-held IRIDIUM telephones transmitting through digital facilities to communicate with any other telephone in the world. Unlike conventional telecommunications networks, the satellite based system will intelligently track the location of the telephone handset, providing global transmissions even if the subscriber's location is unknown.

In areas where compatible terrestrial cellular service is available, the dual-mode handset will provide the option of transmission via the cellular system. Applications for the system vary widely, including business use for persons who must stay in touch with offices located continents away, service for developing nations without telecommunications infrastructure, communications for rescue and supply efforts during natural disasters, and personal use.

26. The atomic number of Iridium is 77 - the number of satellites that were initially going to be used in the system. Now, only 66 satellites are planned, but the name has been retained.

7.2.2 The System

The IRIDIUM network will comprise a constellation of 66 satellites in low earth orbit (also known as LEO), about 420 nautical miles above the earth surface. Compared to geostationary communication satellites 22,300 nautical miles above the earth, the low orbit of the satellites will allow more tightly focused beams to be projected on the ground, ensuring strong signals and communication quality. Echo will be minimized due to the satellites' low orbit, and the receiving antenna can be small enough to be carried on a hand-held subscriber unit.

Small, lightweight satellites (about 1,500 pounds/689 kilograms) will be electronically interconnected to provide continuous worldwide coverage. Communications will be relayed via satellite and through terrestrial gateways, where billing information and user location data will be stored. Service within various countries will be provided through telecommunications authorities and service providers.

7.2.3 The Market

Early wireless industry market estimates suggested worldwide subscription in the low millions at the turn of the century, History has shown, however, that wireless communications usage will eclipse by orders of magnitude those early projections, with cellular utilization reaching as high as 150 million subscribers and paging usage compounded to bring that total in 270 million wireless subscribers.

The IRIDIUM system is uniquely positioned to capitalize on the trend towards wireless telecommunications services by complementing ground-based wireless infrastructures. It will extend the service areas of wireless providers by interoperating with the subscriber's terrestrial cellular network such as GSM 900 and DCS 1800, when available and compatible, and permit access to the IRIDIUM System when alternate transmission techniques are unattainable.

8 Object Database Management Systems

A general introduction to Object Database Management Systems can be found in the book "Object Databases - the essentials" [10], by Mary Loomis. A more detailed discussion, including an overview of a number of implementations, can be found in the book "Object Data Management" [4], by Rick Cattell. A postscript version of "An evaluation of Object-Oriented Database Developments" [11], by Frank Manola, which contains a detailed discussion of the characteristics of Object Databases, is available in the directory /afs/cern.ch/rd45/evaluations in the files odbms_eval1.ps and odbms_eval2.ps.

8.1 The Object Database Manifesto

The Object Database Manifesto[3] is an attempt to define an object-oriented database system²⁷ - an attempt to achieve the equivalent of Codd's original paper[7] defining relational databases. The characteristics of such a system are divided into 3 groups:

- Mandatory: the ones the system must satisfy in order to be termed an object-oriented database system.
- Optional: the ones that can be added to make the system better, but which are not mandatory.
- Open: the points where the designer can make a number of choices.

The mandatory features can be simplified into two rules:

- it should be a DBMS
- it should be an object-oriented system

The first rule translates into five features:

- persistence
- secondary storage management
- concurrency
- recovery
- an ad-hoc query facility

The second one translates into eight features:

- complex objects
- object identity
- encapsulation
- types or classes
- inheritance
- overriding combined with late binding
- extensibility
- computational completeness

8.1.1 Additional features

Whilst largely agreeing with the manifesto described above, we would promote multiple inheritance, distribution and support for versions to the “mandatory” category. Multiple inheritance, for example, is required not only by HEP applications, but also to make CORBA objects persistent in an ODMG compliant database (such objects must inherit from both `d_object`, to provide persistence capability, and from `CORBA::Object`). In fact, it is our view that the database should not impose any artificial constraints upon the user. Thus, in our environment, where C++ is likely to be the language of choice, all features of C++ should be supported by the database.

27. We use the abbreviation ODBMS, as this is favoured by the related standards activities, rather than alternatives such as OODB etc.

In addition to the features described in the manifesto, ODBMSes offer many powerful services, typically including

- schema migration
- versioning of objects
- migration of objects (between versions of the schema)
- data distribution
- data replication
- data caching

8.1.2 Transactions

A database transaction has three basic properties:

- it is application defined. That is, the database itself cannot determine what is and what is not logically a transaction - this is the responsibility of the application,
- a transaction either completes (“commits”) or does not (“aborts”). That is, either all of the associated updates take place or none of them do. Nothing in between is possible,
- once a transaction commits, it cannot be undone (other than by subsequent transactions which reverse the effect of the committed transaction),
- each transaction is executed in an isolated manner. If several transactions are initiated by different users, it is guaranteed that they will have the same effect as if they were executed serially.

Transactions are fundamental to providing consistency and integrity.

In the High Energy Physics environment, transactions may be extremely short, where performance will be the key factor, or very long - perhaps even spanning many processes that execute on different machines.

Object Databases are well-known in markets such as CAD²⁸. For such environments, *long transactions*, where transactions must span many hours or even days, are a necessity²⁹. In our environment, long transactions could be used to process an entire run (or equivalent). Only when processing of the entire run had completed successfully would the transaction be committed to the database, ensuring a consistent view by concurrent readers. (In fact, existing readers would continue to see a consistent state, even after the commit of the transaction.)

Object Databases are now being deployed for production use in the telecoms market³⁰, such as the Iridium project, described earlier in this report (see section 7.2 on page 18). Here, *short transactions* are a necessity, as rapid response is a fundamental requirement.

28. We note that the CAD package in use at CERN, EUCLID, is now using Objectivity as the underlying database technology.

29. In some products, transactions may even span multiple processes and are owned by a user, rather than an individual process.

30. See, for example, <http://www.objectivity.com/apps/Telco.html>, for a list of telecoms applications for which Objectivity has been deployed.

Since Object Databases have been successfully deployed in both of these markets, we can be sure that they will provide good support for both long and short transactions.

8.2 Federated Databases

Cattell[4] defines a *federated database* as one where the data is split across databases from different vendors, or databases with different data models, bridged via gateways³¹.

The federated database concept, as implemented by Objectivity, is what Cattell calls the *conventional distributed database* approach - data is split across multiple databases from a single vendor, using a single data model. This is a mechanism to permit scaling to very large logical databases. The federated database appears to the user as a single, logical, database, sharing a common set of schema and providing transparent, intra-(physical) database references. In the current Objectivity implementation, the federated database can contain 64K physical databases, each of which can in turn contain 64K containers each containing 64K pages (the page size is a federated database constant). However, it is expected that these limits will be raised significantly long before LHC.

8.3 Distributed Databases

Although the computing models that will be adopted by the LHC experiments are only just emerging, it is clear that some level of distribution will be required. Even if a “totally centralised” solution is adopted, distribution would still be used within the LAN - multiple file servers, mass storage servers, database servers etc. will be required, key elements of the system will need to be replicated to avoid both bottlenecks and single points of failure, and indeed the actual processing of the data, from reconstruction and before to data analysis and visualisation and beyond, will be done in a distributed environment. It is therefore important to understand the support provided by Object Databases for such an environment.

We distinguish between a number of different facilities, which are described on more detail below. These categories are as follows:

- Caching
- Distribution
- Replication
- Import/Export

It is important to see the various techniques for supporting distribution as being complementary - each has different advantages and drawbacks and all are extremely important in building a HEP Persistent Object Management System.

8.3.1 Caching

The database automatically maintains an in-memory client cache, where recently accessed and/or created objects are stored. The size of the client cache is typically under the control of the user. The pur-

31. This area is still a subject of much research - specific implementations tend to make a number of simplifying assumptions, such as in the Objectivity case, described below.

pose of the cache is to increase performance - reaccessing objects in the client cache is a much less expensive operation than the corresponding disk access, particularly when network transfer is involved. Database systems often cache complete pages, and hence efficient clustering of objects that will be accessed together is an important factor in determining the effectiveness of the cache.

8.3.2 Distribution

To cope with large databases, and to introduce some level of fault tolerance in the system, it is frequently desirable to store objects in a number of different physical databases, even stored on separate systems. As the total volume of data stored in the system increases, distribution becomes essential. The distribution of objects in different physical databases is analogous to storing different data in different files and filesystems, located on individual storage servers. In the database context, one can arrange the databases into separate *autonomous partitions*³², each of which is capable of working independently of the others. This typically means that some key system resources are replicated, such as the federated database catalogue, and also the lock server. In case of network or system failure, individual partitions can continue to work.

8.3.3 Replication

Replication³³ is an important technique that can be used to improve both performance and fault-tolerance. Replicated objects are objects for which there is more than one *implementation* (cf. "copy"). Typically, the different implementations will be stored on separate servers, often in more than one location. The *degree* of replication may vary within the system - objects that are replicated in all physical instances of a single, logical database are termed *completely replicated*.

In read-only or read-mostly environments, replication offers more performance and fault tolerance advantages than is available with simple distribution of objects. Replication allows data access to be spread over multiple data servers. It reduces network traffic, particularly in the case of WORM³⁴ data, as the data need be transferred over the (wide-area) network only upon write - subsequent read accesses are satisfied via a local database and hence with better performance and lower cost. Fault tolerance is also increased - access to a local replica may continue, even if part of the network is down. Write access is also possible, and controlled by a tailorable voting system. Access to objects remains transparent - it is the responsibility of the ODBMS to select which replica to access and the application should be unaware of which implementation is actually accessed.

Further scenarios are also possible using replication. For example, the individual replicas may be stored in different formats (e.g. corresponding to different machine architectures), or clustered in alternative manners, to satisfy requests with incompatible logical access patterns³⁵.

32. Autonomous partitions may also be used in the case of a single database server.

33. Information concerning replication in ODBMSes is not currently available online, although we do have a non-disclosure copy of the functional specification for the data replication that will be made available in the next version of Objectivity, scheduled for release in Q1 1996. A discussion of the concepts of data replication may be found in <http://www.oracle.com/info/products/symrep/chapter5.html>. See also the discussion in Loomis[10].

34. Write-once, read-many.

35. This functionality, although potentially extremely interesting, will not be available in the forthcoming release of Objectivity.

Replication will clearly be one of the most important distribution techniques and is high on the list of priorities for future extensions of the ODMG standard, discussed below.

8.3.4 Import/Export facilities

Databases may be “detached” and “reattached” to the federation, or explicitly exported or imported. This can be useful for mobile computing, or for example when a remote copy of a database is lost. Although these techniques will be useful, it is our feeling that replication offers much more functionality and is a significantly more attractive approach than import/export, which corresponds more or less to the current techniques of data export using boxes of Exabytes.

8.4 Schema Evolution

Schema evolution operations include changes to classes, class contents and relationships between classes, such as associations, references and inheritance. In an Object Database environment, changes are made to the schema by modifying the corresponding Object Definition files and then processing these files using the appropriate database tool. Schema evolution is extremely flexible. For example, it allows you to perform the following operations:

- delete a data member
- add a data member with system default or user defined default values
- rename a member
- change the type of a data member
- reorder data members
- add/change/delete method definitions
- change the size of array data members
- add/delete/replace a base class
- add/delete/change associations or references
- change the cardinality of references
- etc.

In addition to changes to the database schema, the affected objects, i.e. the persistent instantiations of the schema, may be converted in a number of ways, e.g.

- immediate - all affected objects within a federation are changed using a special upgrade application
- deferred - affected objects are changed only when accessed. This has the advantage that only those objects that are used are changed, but does mean that access times can be unpredictable during the conversion period
- on-demand - affected objects in specific containers, databases, or within the entire federation are updated upon user request

It is clear that the data models of the various LHC collaborations will evolve with time. Support for such evolution by the database provides a very powerful and convenient mechanism for coping with such change.

8.5 Data Access

We describe below a number of techniques that could be useful for providing efficient data access. It is unlikely that any given method will be the most efficient for all types of access, and an important component of the prototyping that is being performed in the framework of the RD45 collaboration is to try to understand the suitability of the various techniques. What is described below is specific to the Objectivity ODBMS, but similar features are available in other products.

8.5.1 Naming Objects

Objects within a database may be named, and therefore accessed, by a character string. Individual objects may have multiple names, within different scopes³⁶. One could, for example, have a named object called *Higgs_Candidate*, which could then be used to access individual candidate events via associations, described below. Similarly, one could locate individual runs (run objects) by name, and use associations to access events within the run in question. Names could also be used to identify sub-detectors, e.g. within the scope of individual events.

8.5.2 Associations

Associations are logical links between different objects. They may be uni- or bi-directional, and may have cardinality of one-to-one, one-to-many, many-to-one or many-to-many. Associations, including their direction and cardinality, must be defined in the class definition files, specified using the Object Definition Language (ODL). In the case of Objectivity, associations are implemented using classes, with methods of their own, that are automatically provided by the ODL processor.

8.5.3 Iterators

Iterators allow one to access individual objects that meet certain criteria. For example, one can iterate over objects

- of a particular class (and subclasses)
- below a certain level in the storage hierarchy (e.g. all objects in a federated database, in a physical database, in a container)
- within an association
- that match specified index keys
- that meet a specified predicate query

The following steps are required to use an iterator:

- The iterator must be declared
- It must be initialised
- The **next()** member function is then used to visit all appropriate objects

For example, one could use an iterator to visit all track objects with an association to a specific vertex object, all events within a run etc.

36. Thus, an individual object may be referred to as “father” within one scope, “son” in another, and “brother” in a third.

8.5.4 Indexing

Indexing is a technique to improve performance. Objects may be indexed on data members, and retrieved according to exact or range match of attribute values.

9 Object Database Management System predictions

A standard for Object Database Management Systems, known as ODMG-93, has been developed by a group of vendors - the Object Database Management Group. The primary goal of the ODMG is to put forward a set of standards that permit an ODBMS customer to write portable applications, i.e. those that can run on more than one ODBMS product. All members of the ODMG are committed to supporting this standard, by delivering standards-conforming products.

9.1 Status of the ODMG standard

The ODMG-93 has gone through 2 revisions since its initial release in 1993 - versions 1.1 and 1.2, the latter being released in November 1995. The next version of the standard, version 2.0, is scheduled for release at the end of 1996/early 1997. The ODMG works very closely with related standards bodies, such as the OMG and X3H2 (SQL3). Indeed, the work of the ODMG has already influenced some of these standards, for example the OMG query service and persistent object service. The ODMG is currently preparing a certification suite for use in verifying if commercial products do indeed conform to the standard. So far, 9 vendors have announced plans for some level of ODMG support (either version 1.1 or 1.2 of the standard for one or more components) in products that will be delivered during 1996.^{37 38}

Version 2.0 of the standard will contain a number of enhancements which are of interest to HEP. The complete list of priorities for V2.0 are listed in Appendix B (see section 15 on page 41), in order of descending priority. Although it is unlikely that all of these items will be completed by the end of 1996, there is a strong correlation between the priorities of HEP and the ODMG.

Essentially, we can assume that almost all of the items in this list will appear in commercial products by 1998³⁹. Some are already supported in a non-standard fashion, e.g. the ASCII database input/dump format is currently supported for Objectivity/DB by a product from Micram, schema meta-model read access, security, schema evolution etc. will all be available in products shipped in 1996. Further out, we can expect “plug-and-play” between ODMG-compliant ODBMSes and CORBA-compliant ORBs, distributed transactions spanning databases from different vendors⁴⁰, and, most importantly, convergence between the OQL and SQL3 standardisation efforts. We note that in many of these areas, one of the key players involved is Objectivity.

37. The ODMG has defined the terms ODMG-compliant and ODMG-certified for ODL, OQL and C++ and SmallTalk-bindings. Certification requires that the product in question has passed the ODMG certification suite.

38. Unfortunately, the information regarding which products will support which binding can not currently be distributed outside the ODMG, but should be made public during the course of 1996.

39. The exception is the preprocessor-based C++ binding, which is listed for historical reasons only.

40. Some vendors are keen to provide just such as demonstration at a future Object World - perhaps even OW96!

The list of priorities for post-V2.0 of the standard will not be finalised until the end of 1996. However, we have already requested that replication be added to the list. Beyond that, we expect support for interfacing for mass storage systems to be an item of importance. Recently, the idea of a Java binding has been raised, and an initial working group meeting will have been held by the time that this report is published.⁴¹

At this stage, it is uncertain how long the momentum of the ODMG will continue⁴². We believe that it is safe to assume that V2.0 of the standard will be completed, and that several, if not numerous, conforming products will be available within 1-2 years of its publication. Beyond that, at least one more revision is likely, appearing perhaps in late 1998 with conforming products no later than the year 2000. Although there are no plans to disband the ODMG after a certain release, we do not feel that it would be wise to base LHC computing models on features which are not yet in the wish-list for post-V2.0 versions of the standard or are not in the product plans of the vendors today. Fortunately, we do not see this as being particularly constraining - existing products are already highly functional and the plans for the next year and beyond are impressive.

9.2 Very Large Databases (VLDB)

Databases with a size of several tens of GB to hundreds of GB exist in production today. The size of databases can be expected to increase steadily over the coming years, reaching 1 TB and beyond. There is general consensus in the industry that truly large databases will be realised using multiple physical databases, or very many large databases (VMLDB)[24]. Predictions from the IBM Database Technology Institute at the IBM Almaden Research centre suggest that "Databases in the year 2000 will need to provide support for massive amounts of data, upwards of 100 terabytes." [25] The Gartner Group predicts that databases sizes will approach 1 PB as early as the year 2000 and that they will be achieved as follows:

- Database architectures comprised of multiple DB engines accessed through a single layer
- Database parallelism
- Distributed and Replicated data across multiple servers
- Mainframe usage as a viable server platform for the future(!)

The use of replication is cited for a number of reasons, including:

- Populating data warehouses
- Remote backup and fault tolerance
- Mobile computing and support of disconnected clients
- Local access of information
- Load balancing between multiple machines

41. In fact, Rick Cattell, the ODMG chair, has recently moved from SunSoft to Java Products Business, also part of Sun, which would suggest that a Java binding is likely. If such a binding were produced, persistent Java objects could then interoperate with persistent C++ and Smalltalk objects in an analogous fashion to what is currently possible via the existing language heterogeneity support.

42. It is planned to hold an ODMG meeting at CERN, perhaps as early as July 1996.

O₂, in the book “Objets et bases de donnees” [13], predict that database sizes in the range of 10¹⁶ bytes (10 PB) will exist in the early years of the next millennium, and that such problems will be solved using distributed database architectures and other such techniques.

Objectivity are involved in a number of projects world-wide, where highly reliable, distributed, very large databases are required. These include the Motorola Iridium⁴³ project, the Sloan Digital Sky Survey⁴⁴ and of course a number of projects in High Energy Physics.

9.3 Integration of ODBMSes with Mass Storage Systems

Although we predict that disk farms with a capacity of 1 PB or more may be possible by 2004, we nevertheless require a coupling of Object Databases with Mass Storage Systems for various prototyping activities, and indeed as safety precaution should disk technology not evolve sufficiently rapidly as to make multi-PB disk farms both technically possible and affordable. There are a number of techniques by which Object Databases could be coupled with Mass Storage Systems. These are discussed below:

- Using a migrating filesystem.

This is the simplest solution - indeed it requires no work, as it is in production use today! As Object Databases use filesystems in which to store databases (they are seen by the filesystem as ordinary, if rather large, files), Object Databases may be used transparently with migrating filesystems. It remains to be seen whether future parallel filesystems will be integrated with mass storage systems, such as ADMS and/or HPSS, in a sufficiently clean manner as to make this approach viable. Given the absence of a production migrating filesystem at CERN, this approach is not applicable for the short-term.

- An explicit interface at the application level

In this case it is the responsibility of the application to perform the necessary interaction with the mass storage system. Although this approach is certainly viable, some of the transparency of using databases is lost, as application specific methods must be used to access and navigate through the data.

- An explicit interface at the database level

In some senses, this is just a variation of the above. By trapping database open failures, one can transparently interact with the mass storage system and retrieve physical databases as required. The main advantage of this scheme is that it retains the database transparency that is lost by the previous approach. In addition, it is not tied to a specific mass storage system - by writing/modifying the appropriate handler, site specific details can be catered for.

- A tight coupling at the level of database pages (or containers)

43. See <http://www.ee.surrey.ac.uk/Personal/L.Wood/iridium.html> for various pointers to the Iridium project. The choice of Objectivity as the database system for this project gives us confidence that the product will have a long life.

44. See <http://www-sdss.fnal.gov:8000/> for information regarding the Sloan Digital Sky Survey.

In this approach, the ODBMS stores/retrieves unused pages in/from the mass storage system. In our opinion, it is the least attractive of the various alternatives. Firstly, it requires a tight coupling of the database with mass storage technology. In the absence of a standard API for mass storage systems, this coupling will inevitably be system specific. In addition, the size of database pages, typically 4096 bytes, is a very inefficient quantity for moving to/from mass storage systems, where a chunk of tens of MB at least is preferred.

9.3.1 Integration of Objectivity with Mass Storage Systems

We anticipate that sample code showing how to handle the database open failure will be supplied by Objectivity during 1996. This sample code is likely to be used by NA45 in their CS-2 based production run. As a number of other large sites are considering the use of Objectivity (FNAL, LBL, LLNL), we expect that the necessary support will be provided as part of the base product in the 1997/1998 timeframe. It is also our intent to add this requirement to post-V2.0 work on the ODMG standard (V2.0 will be published early in 1997).

10 Standards integration

Commercial solutions based on the standards described above already provide a great deal of interoperability. Object Databases are able to work together with Object Request Brokers and Object Request Brokers are able to interoperate with OLE based applications. ORBs themselves may interoperate using either Internet protocols, or other protocols such as DCE⁴⁵. The interoperability of these individual components is either already standardised, or will be incorporated into a future version of an existing standard. Full interoperability of standards-conforming components can therefore be expected within 2 years at the latest. The interface to mass storage systems, for example via the DMAPI, is also being standardised. Future filesystem work, such as an enhanced parallel filesystem from IBM, includes an interface to mass storage systems such as HPSS and/or ADSM.

11 LHC Computing Model scenarios

In this section, we present a number of scenarios that are currently being discussed within the ATLAS and CMS Computing Model working groups and discuss how these models could be implemented using a combination of Object Database and Mass Storage Systems.

Some driving considerations behind the computing models are

- transparent access to any component of the data must be provided worldwide to all members of the collaboration
- the data should nevertheless not be moved without good cause

45. In fact, different protocols, including also proprietary protocols, may be used within separate ORB domains.

11.1 The Fully Centralised Solution

In the fully centralised model⁴⁶, all of the data at all stages of processing resides at a single site, presumably CERN. The data resides in an ODBMS, which provides transparent access to the needed data, i.e. selected (sub-)events. Little used data is migrated transparently to a mass storage system and recalled upon demand.

Client-server technology is used to provide remote data access. In addition to the data, the bulk of the processing power is also installed at the same central location. This is used not only for reconstruction, presumably done “on-line”, but also for data selection - only the selected data is transmitted over the network to the users’ workstation for visualisation, minimising network bandwidth requirements.

It is important to point out that even the fully centralised solution is, in fact, distributed. The reconstruction will be done, presumably in parallel, on a farm of workstations or PCs, the data will be stored in multiple physical databases that will reside on a large number of physical volumes and be managed by many database/fileservers. It is also highly likely that replication will be employed, both for important resources, such as lock servers (to avoid single points of failure and to reduce bottlenecks), as well as frequently accessed data, for load-balancing purposes⁴⁷.

Although it is implicit in the above, it is important to point out that a large-scale mass storage system is only required at a single, central, site. As is true also for the CPU farm, the operational costs of a single, large system are not significantly larger than, for example, each of the 3 systems one third of the size, that would be needed if the rawdata were distributed to 3 sites for processing. Thus, this solution will offer HEP-wide savings, which are likely to be significant, in terms of man-power.

11.2 The Partially Distributed Solution

In this solution⁴⁸, the bulk of the data, i.e. the rawdata, is still kept centrally. Some of the data, however, is distributed (replicated) to a number of regional centres. The data available at each such centre will typically correspond to event selections, depending on the type of analyses being performed at the site in question. Normally, the rawdata is not available (locally) at such sites - it can, of course, be transparently accessed from anywhere within the collaboration, although this involves transferring the necessary data over the network from the central site. The data made available at such regional centres corresponds to that subset of the data needed for typical analyses, including a simple event display. It is important to stress that, in contrast to today’s situation, data which is not stored locally can still be accessed transparently using database pointers. Thus, if it is discovered that the subset does not contain all of the data needed for a given analysis, it is not necessary to recreate a new selection - the missing data may simply be accessed over the network. If accessed frequently, the extra data may subsequently be replicated to the regional centres in question.

46. See, for example, <http://hepunix.rl.ac.uk/atlas/cmos/scenarios/centralised.html> for a discussion of a centralised proposal by Steve Fisher/ATLAS.

47. In some senses, therefore, the differences in the models discussed in this paper are only in the degree of wide-area distribution that is employed, and not in the underlying technologies.

48. See, for example, the proposal by Krzysztof Sliwa/ATLAS in http://atlasinfo.cern.ch/Atlas/GROUPS/CMOS/SLIDES/sliwa_111095.html.

As suggested above, the most efficient technique for making such subsets available at regional centres is *replication*. Rather than make explicit copies of files or tapes, which are then exported by hand over the network or via magnetic tape, the appropriate data is automatically stored in multiple locations by the database system itself. The network bandwidth required obviously depends on how much of the data is replicated, and this technique will almost certainly be impractical if the full data sample, including the rawdata, is to be replicated in the wide area. However, if the data can be logically divided into separate “streams”, e.g. corresponding to different event categorisations/physics processes, and only a subset of the processed (reconstructed) data is replicated, only modest network bandwidths are required. For example, to replicate 10 KB/event (1% of the rawdata volume) for a “stream” corresponding to 1% of the events coming out of the level 3 trigger⁴⁹, only 10 KB/second is required to keep up with the rate of data acquisition. Although 10 KB/event is somewhat less as a fraction of the rawdata volume than is common today⁵⁰, it is again important to stress that the database approach allows one to define a smaller subset than would otherwise be possible - all of the data corresponding to individual events remains transparently accessible, as opposed to today’s situation whereby navigation from an event summary to the complete information is non-trivial. In addition, it is clear that there are some streams, such as Higgs candidates - where only just a few candidates per collaboration per year are expected - that correspond to significantly less than the 1% figure quoted above.

As is true also of the fully centralised scenario described above, this model only requires bulk mass storage at a single site - external sites need only provide relatively modest disk farms. This scenario is also likely to generate significantly less network traffic, although this will require extensive modelling to verify.

11.3 Distribution of Rawdata

In this scenario, the rawdata is distributed to a small number of regional centres. The processed data for any given event resides at the same site as the rawdata. This scenario differs from the fully centralised scenarios in a number of important respects:

- the physical distribution of the rawdata to remote sites results in an unavoidable delay in making the processed data available to the collaboration
- extra complexity in terms of book-keeping and data processing is implied
- extra overhead (the physical export/import of the data volumes)
- extra cost (the hardware required at both ends to export/import the data)

Although it has been argued that this solution makes better use of existing resources, such as existing remote computer centres, this argument is not entirely water-tight - it is doubtful if any of the existing hardware will be deployed in the era of LHC data-taking.

It is our conclusion that this solution offers no obvious advantages, but numerous disadvantages, including increased cost and complexity.

49. Which is assumed for these purposes to level 1 MB events at 100 Hz.

50. The reduction factor on the FNAL Collider experiments is typically 10-20.

11.4 Persistent Storage versus Recalculation of “DST” information

In this scenario, only the rawdata and a compact summary of the key features of each event is stored persistently. Physics analysis selects events according to their key features, and then the required information is recalculated using the latest calibrations and algorithms.

This can be simply achieved in a database environment in the following fashion.

- only the rawdata is written to the data store, in multiple physical databases which are written into a mass storage system
- the “key features” of each event are stored in an Object Database (tag objects)
- physics analysis selects events by issuing a query, either in a programming language such as C++, or using a query language such as SQL3/OQL, against the tag database
- the query returns a collection of matching tag *objects*, each with its own *methods*
- the user then iterates through the collection, using methods provided on the collection object
- access to components of individual events is performed using methods on the individual tag objects
- these methods restore the appropriate rawdata from the mass storage system if required, and run the appropriate methods to perform reconstruction, using the latest calibrations and algorithms
- the reconstruction may be performed on all data in a given physical database, or preferably only on the events accessed
- in the latter case, only those events that are referenced will be processed, improving efficiency
- the databases, which previously contained only the rawdata, now also contain processed data for those events which have been accessed, and reside in a cache until removed by a garbage collection
- the accessor methods (on the tag objects), can be coded so that the reconstruction is only redone if calibrations and/or algorithms have changed, rather than upon each access, significantly improving efficiency in the case when the databases remain in the cache for a reasonable period of time

A number of variations on this theme are possible. For example, one could also store the reconstructed data into the mass storage system and use the database accessor functions to reprocess as required. Indeed, the use of accessor functions would make different strategies possible for different (categories of) events - if the reconstructed information was missing or invalid, it would be added automatically, transparently.

11.5 Access to complete events

In all of the scenarios described above, the rawdata resides centrally⁵¹. Nevertheless, one must be able to access complete events. In the case of access to single events, e.g. for event display purposes, the bandwidth required is not an issue - 1 MB of data can be transferred even over relatively slow lines with tolerable delay. If large numbers of complete events are required, e.g. for reconstruction, then it clearly makes sense to perform the processing close to the data. In the case of reconstruction, this is well understood issue and does not present a problem. For other types of access, it would be preferable if the system were able to perform some “cost-estimate” of the query, and automatically

51. It is, of course, possible in any of the scenarios for “exceptions” to be made, and for samples of rawdata to be distributed, as required. This cannot, however, be the norm, due both to network and storage constraints.

run the task close to the data if required. Simpler solutions, such as using access control to deny access to rawdata objects from nodes outside CERN, may be sufficient to avoid such problems.

11.6 Access to sub-events (sub-detector)

Access to complete sub-detector information can be expected from sites with a responsibility for a given sub-detector. In addition to the simple solution of running data-intensive queries centrally, as above, the appropriate data may also be replicated to the relevant institutes. However, given the fact that these data will be accessed relatively infrequently, and by a well-defined set of people, e.g. for sub-detector studies and/or calibration purposes, there is no obvious advantage in replicating the data.

11.7 Access to individual components of an event (“Ntuple” style)

The most frequent style of access to the data is expected to be for physics analysis - the equivalent of what is done today using Ntuples. Here, the database approach offers a number of significant advantages:

- there are no arbitrary constraints on the data model - the experiment and/or user is at liberty to define their own data model and not conform to that of the analysis framework
- the data that may be accessed is not limited to a specific subset (the Ntuple “columns”). *Any* component of *any* event may be accessed, although, for efficiency reasons, it will certainly make sense to cluster together the most frequently accessed parts of events⁵² (which may vary according to logical streams). Thus, one may perform an event display of all events in a given histogram bin, without requiring that the “Ntuple” contain a copy of the data need for the event display, but simply by exploiting transparent database access to the needed data
- the use of replication provides a scalable solution to wide-area data access

12 Capabilities of a HEP Persistent Object Management System

High Energy Physics has for long been quasi-unique in terms of the vast quantities of data that are generated, and in the style of access - fully distributed - that must be supported. Our requirements have traditionally out-stripped what has been possible with standard and/or commercial solutions, and this has led to the development of many HEP-specific solutions to the problem of data management and access. Many examples of such solutions can be seen in use today at LEP and elsewhere. The requirements of the next generation of HEP experiments, which are currently being discussed in the Computing Model Working Groups of ATLAS and CMS, are even more daunting than those of today’s experiments. In addition to a significant increase in the volume of data that must be handled, we are facing a major shift in paradigm - from the traditional, Fortran-based procedural approach, to the Object-Oriented one. Fortunately, we have both time and technology on our side. Recent years have seen dramatic changes in computing technology, and current trends are likely to continue or even accelerate.

52. Indeed, appropriate clustering of data is one of the key techniques available to achieve performance. Investigation of clustering techniques will be a very important part of our future research work.

Data processing in HEP has traditionally been based upon sequential access, largely for historical reasons as the bulk of the data was stored on sequential media (magnetic tape). Over the years, various techniques have been invented to provide differing degrees of random access to data. Increasingly, the concept of “database-computing” has been discussed - indeed, many techniques, borrowed from the database industry, are in production use in HEP today.

The technology and trends described earlier in this report clearly indicate that an extremely powerful system can be built out of industry-standard hardware and software components. We will see powerful, distributed database systems, exploiting also CORBA and other technologies, in wide-scale production over the next few years⁵³, and the successful deployment of such systems can be expected to result in even more powerful technologies that will be available sufficiently early as to benefit experiments at LHC. Nevertheless, a great deal of work still remains to be done. The various scenarios discussed in the Computing Model Working Groups must be modelled, and critical components prototyped. The early deployment of some of these ideas, such as on the NA45 experiment, will help us to understand better the technologies involved, and further refine the models.

Given the information presented above, we believe that it is clear that an extremely powerful data management solution can be built, that will not only greatly exceed the functionality of existing solutions, but will also offer clean integration with the programming environment, and be sufficiently modular (and standard) as to permit adoption of new technologies, as they appear over the lifetime of LHC. We believe that by working closely with the appropriate partners, a solution appropriate to the management of one of HEP’s most precious resources - the data - can be achieved.

We believe that LHC-era computing will be database computing, and in particular parallel, distributed, object database computing. We describe below how the database system could be used as an integral component of the entire production chain, from raw data to publications. Such a system will make the famous quote from CHEP-92⁵⁴ reality, and no longer a dream.

Although a system based on Object Databases and Mass Storage Systems will offer a great deal of flexibility in terms of data organisation, computing models variants etc., we have chosen to focus on a specific scenario, namely the partially distributed scenario, both for clarity and because this currently appears to offer the best combination of functionality and performance.

12.1 The Reconstruction Process

It is widely accepted that the reconstruction process will be done in parallel. Here, we already see benefits from a database approach [27] [28] - parallel update requires database-like features if integrity is to be provided. In the RD45 model, the reconstruction phase will also be critical for data placement - the data must be correctly “placed” (e.g. like data clustered together physically on the same or adjacent database pages, in containers, in physical databases etc.) and the appropriate mechanisms for rapid and transparent access to the data built. At this stage, it is not clear which techniques will be most suitable, and indeed different approaches might be better in various cases, but it is clear that

53. Such as the Motorola Iridium project, which uses both and ODBMS (Objectivity) and an ORB (ORBIX).

54. Transparent access to any byte in a petabyte.

powerful database techniques will be required to ensure that highly efficient data access is possible. The input data to the reconstruction process is time-ordered. The output data, both the rawdata and the added information, will be organised by event characteristics, offering significant advantages for subsequent data access. Time-ordered access to the data will still be possible, but there will be no need to retain a copy of the data that is specifically organised in that way.

Using a combination of parallel filesystem/database technology, single stream rates in excess of 100 MB/second should be easily achievable. The transfer of data into the mass storage system will also take place at similar rates, probably again employing parallelism, to a small number of drives.

The reconstruction process, in addition to adding the reconstructed information to the appropriate databases, will update the *event tag* databases. The latter databases will be both *streamed*⁵⁵ and *replicated*. Both of these techniques will improve data access. Streaming will as the primary selection will have to query smaller samples of data. Replication will help in reducing bottle-necks and lowering network traffic - queries from remote sites will typically be satisfied against a replica of the event tag database that resides locally. As the size of the tag objects is expected to be small - perhaps as low as 100 B, the data rates required to replicated these objects to remote sites is also very small. A further saving could be obtained by only replicating certain streams of the tag objects, but in any case the required bandwidth is sufficiently modest as to not pose a problem.

Replication of parts of the event data will also be performed. Here, carefully modelling and prototyping is required to understand the trade-offs between caching of the needed data on demand versus replication and in helping to define appropriate subsets of the data. This work clearly requires close collaboration with the experiments themselves.

It will be possible to perform the reconstruction of an entire “run” (or equivalent) during a single (in fact, nested) transaction, even if it is performed by multiple processes in a distributed environment. The final “commit” at the end of the transaction would write-out any remaining pages from the client caches and update internal database pointers. In the case of reprocessing, users could be accessing existing data and retain a consistent view even whilst the reconstruction was in progress - the previous version of the data would be seen until the transaction responsible for the reprocessing committed. In fact, even beyond that point, existing read transactions would continue to see the previous data. As the data will be stored in a database, it would be possible to reprocess only parts of the event, updating the information in place, e.g. when new calibrations or algorithms became available. It would even be possible for the database to “control” the entire production process, and automatically reprocess individual events and/or subevents on demand. This, however, would imply the use of special accessor functions.

12.2 Data Analysis

Data analysis would work directly with the database. The database query mechanism, either in C++ or OQL/SQL3, would deliver the appropriate data to the visualisation system. No special reformat-

55. In this case, the streaming may in fact correspond to the use of different containers within the same physical database(s), as opposed to streaming into different physical databases, as is likely for the data itself. In a future release of Objectivity, it will be possible to store containers in separate files, as is done today for physical databases, so the distinction may, in any case, disappear.

ting of the data, change of data model (e.g. conversion to Ntuples etc.) would be required. Access to additional information regarding the event would be transparent, although there would of course be an additional cost if parts of the data, e.g. the raw data, were accessed. However, it would nevertheless be transparent and fully integrated.

Care would be required to ensure that queries were executed “close” to the data. That is, either on local replicated or cached copies of the data, or “sent” (again, transparently) to an appropriate data warehouse. The required techniques to perform such operation exist and are in production today.

12.2.1 An Example Data Analysis Framework

To explain how an Object Database may be used in a data analysis environment, we describe a possible scenario where the data is stored in an ODBMS, and the analysis/visualisation is performed using a system based upon the data-flow model, such as Iris Explorer⁵⁶.

In this system, an application is built up out of individual modules, connected together via data-flow pipes⁵⁷. Typically, one module will read input, e.g. from a file, then pass it to another module which make perform some transformation, and finally pipe the data into a module which renders an image⁵⁸.

The user would type the appropriate “SELECT” statement into a panel, and this would be transferred to the reader module. This module would execute the corresponding database query and thus select the appropriate (sub-)objects from an ODBMS. To avoid unnecessary network traffic, the reader module should preferably run on or “close to” the dataserver. The output of the reader module, typically a small fraction of what is read (only the selected components of selected events), would then be transferred to the next module in the pipeline, which might perform some transformation or cuts that could not easily be formulated as a database query. Note that the ODBMS only reads those pages that are necessary to perform the selection, and only the data that satisfies the query is transmitted to the next stage of the pipeline, optimising both disk and network traffic. Finally, the data would be passed (by reference) to the appropriate render module, which would typically display a histogram or scatter plot. Each module in the pipeline may run on the same or separate processors, either in the local or wide area. The database is responsible for delivering the needed data; the visualisation framework does not force a particular data model (e.g. Ntuples⁵⁹) on the user; the system is extensible and configurable - extra modules may be added as required, modules may be changed and recompiled dynamically; the system exists today.

56. See http://www.nag.co.uk:70/Welcome_IEC.html for more information on Iris Explorer. Other systems built upon the data-flow model include IBM's Data Explorer.

57. The data does not actually flow down the pipes - only references are sent, unless the modules run on separate machines, in which case TCP/IP sockets are used.

58. Much more sophisticated applications are, of course, possible.

59. Although, for performance reasons, it may be necessary to physically cluster the data required for analysis, we wish to stress that it will *not* be necessary to use a different object model for these data, and certainly not one imposed by the visualisation system.

12.3 Alternative data analysis frameworks

Given the recent explosion of Web technology, the current excitement over Java⁶⁰ and the likelihood of a future Java-binding to ODMG (see section 9.1 on page 26), some people predict that data analysis might occur using advanced Web browsers and Java objects - data + applets - which perhaps even permit data analysis to occur from \$500 Internet PCs!

It is clearly too early to reliably predict what technology will exist in this area in the LHC data-taking era, but a reflection on the sometimes stunning progress over recent years suggests that a revolution, even more dramatic than the change from punched-cards to today's sophisticated GUIs. is by no means excluded.

12.4 Other database applications

We expect that databases will be used for many purposes other than the storage of event data. Such applications include:

- calibration data,
- detector configurations,
- run logs,
- production control.

It may even be appropriate to use databases for code management. The question of ensuring that the appropriate libraries - containing the methods of the objects stored in the databases - are installed on all platforms will certainly have to be addressed.

However, none of these applications has requirements that are anything like as stringent as those pertaining to event data management, and could certainly be met by a system capable of handling the event data itself.

13 Conclusions

We have seen that the trends for storage, both hardware and software, and for Object Databases look very promising. We are confident that they will allow us to build an extremely powerful and flexible computing environment to meet the challenges posed by the next generation of High Energy Physics experiments at CERN and elsewhere. Our extrapolations have been made according on a conservative basis - a system with the capabilities described above, but on a slightly smaller scale, would be easily achievable for pre-LHC experiments, such as BaBar, Fermilab run II, or experiments at RHIC. Early demonstration of many of the features described in this report will be made during the coming months.

60. See, for example, http://java.sun.com/whitePaper/javawhitepaper_3.html.

14 Appendix A - list of key HPSS features

14.1 Implementation Requirements

- C programming language
- UNIX platform
- No required kernel modifications
- National Language Support (NLS) Enabled

14.2 Design Standards

- Client / Server based
- Guided by IEEE Mass Storage Reference Model: version 5
- Layered architecture to enable plug and play components
- Use of POSIX, when available
- Open Software Foundation Distributed Computing Environment (DCE) infrastructure

14.3 Scalability

- Storage capacity to multiple petabytes
- Throughput per transaction of multiple gigabytes per second
- Bitfile sizes up to 2^{64}
- Name space entries
- Number of simultaneous users
- Introduction of additional devices
- Software striping
- Distribution and multiprocessing of servers

14.4 General

- Support concurrent client accesses
- Separation of control and data
- Reliable storage of system data and file metadata
- Transaction Management - commit or abort transactions to maintain consistent metadata

14.5 Interfaces

- Client API (both sequential and parallel, based upon POSIX file and I/O functions)

- Access to data via random access lists
- Third party copy
- Standard File Transport Protocol (FTP)
- Network File System (NFS) V2
- Parallel FTP
- IBM SPx PFS
- Distributed File System (DFS) / Andrew File System (AFS) (future)
- Virtual File System (VFS) (future)
- NFS V3 (future)
- DMIG (future)

14.5.1 Media Support

- 3480 / 3490 tape
- Ampex D-2
- IBM NTP
- SCSI / SSA attached disk
- IBM 9570
- Support of network and directly attached devices (multiple storage hierarchies)
- Maximum Strategy GEN-5
- Others (future)

14.5.2 Robotics Support

- Ampex DST 800
- STK robotics
- IBM 3494 / 3495 robotics
- Others (future)

14.5.3 File Management and Storage Management

- Maintain POSIX name space and file characteristics
- Migration and caching (including partial)
- Multiple dynamic storage hierarchies
- Undelete
- Software striping
- Repacking

14.6 System Management

- GUI based (X Windows using Motif)
- System and server configuration
- Reporting and monitoring of managed objects, alarms and events
- Perform management operations (add volumes, mount / dismount drive, import / export volumes, etc.)

14.7 Security

- Uses DCE security
- Provides Generic Security Service (GSS) interfaces for server / user authentication
- Provides authorization checking for servers / users
- Provide enforcement of bitfile permissions & simple audit
- ACL support
- Mandatory Access Control / Security labels (future)

14.8 Other Features

- Accounting
- Audit physical volumes
- UniTree migration

15 Appendix B - list of priorities for ODMG V2.0

- Schema meta-model objects (read-only)
- Schema meta-model objects (read-write)
- Distributed transactions (XA, OMG protocols)
- ASCII database input/dump format
- ORB Object Database Adapter
- Define ODL -> C++ mapping
- More marketing activities
- Schemas and operations that span distributed databases
- Security and authorization model
- Chapter 2 consistency and elaboration
- Schema evolution -- instance migration
- Compliance test suite
- Explicit locking calls
- Better define C++ <-> SmallTalk interoperability
- Work with X3H2 to merge OQL and SQL3
- C++/SmallTalk ORB requirements: include, external OIDs
- Database Administration interface
- Rigorous basis for Chapter 2
- General OMG positioning/interoperation
- Nested transactions in Object Model, C++, SmallTalk
- Versioning and configuration model
- Keys and extents in C++, SmallTalk
- OMG Collection Service
- OQL operator for queries not on subtypes
- Position SQL3 as another binding
- Multi-attribute keys
- Define preprocessor-based C++ binding

16 Glossary

ADSM - A storage management product from IBM

AFS - the Andrew (distributed) filesystem

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

CORE - Centrally Operated Risc Environment

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

DMIG - the Data Management Interface Group

GB - 10^9 bytes

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

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

IEEE - the Institute of Electrical and Electronics Engineers

MB - 10^6 bytes

MSS - a Mass Storage System

NFS - the Network Filesystem, developed by Sun

ODBMS - an Object Database Management System

ODMG - the Object Database Management Group, who develop standards of ODBMSes

OMG - the Object Management Group

OQL - the Object Query Language defined by the ODMG

ORB - an Object Request Broker

OSM - Open Storage Manager: a commercial MSS

PB - 10^{15} bytes

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

SSSWG - the Storage System Standards Working Group

TB - 10^{12} bytes

VLDB - Very Large Database

VMLDB - Very Many Large Databases

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

17 References

- [1] Computing at CERN in the LEP era
- [2] Computing at CERN in the 1990s
- [3] The MUSCLE Report (The Computing Needs of the LEP Experiments)
- [4] Object Data Management. R.G.G. Cattell, Addison Wesley, ISBN 0-201-54748-1
- [5] DBMS Needs Assessment for Objects, Barry and Associates (release 3)
- [6] 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 [8].
- [7] A relational model for large shared data banks, E. F. Codd, Communication of the ACM, Volume 13, Number 6, (June 1970), pp 377-387.
- [8] Building an Object-Oriented Database System: The Story of O₂. F. Bancilhon, C. Delobel, and P. Kanellakis.(eds.) Morgan Kaufmann, 1992.
- [9] Object Oriented Databases: Technology, Applications and Products. Bindu R. Rao, McGraw Hill, ISBN 0-07-051279-5
- [10] Object Databases - The Essentials, Mary E. S. Loomis, Addison Wesley, ISBN 0-201-56341-X
- [11] An Evaluation of Object-Oriented Database Developments, Frank Manola, GTE Laboratories Incorporated
- [12] Modern Database Systems - The Object Model, Interoperability and Beyond, Won Kim, Addison Wesley, ISBN 0-201-59098-0
- [13] Objets et Bases de Donnees - le SGBD O₂, Michel Adiba, Christine Collet, Hermes, ISBN 2-86601-368-9
- [14] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1, 1991.
- [15] Object Management Group. Persistent Object Service Specification, Revision 1.0, OMG Document numbers 94-1-1 and 94-10-7.
- [16] The Object Database Standard, ODMG-93, Edited by R.G.G.Cattell, ISBN 1-55860-302-6, Morgan Kaufmann (publishers).
- [17] Magnetic and Optical Data Storage - Tutorial Notes from the 10th IEEE Symposium on Mass Storage Systems, May 1990
- [18] "Crisis in Mass Storage" - Proceedings of the 10th IEEE Symposium on Mass Storage Systems, May 1990
- [19] Storage System and Design Issues - Tutorial Notes from the 11th IEEE Symposium on Mass Storage Systems, October 1991
- [20] "Distributed Storage Environments" - Proceedings of the 11th IEEE Symposium on Mass Storage Systems, October 1991
- [21] "Putting all that Data to Work" - Proceedings of the 12th IEEE Symposium on Mass Storage Systems, April 1993
- [22] "Towards Distributed Storage and Data Management Systems" - Proceedings of the 13th IEEE Symposium on Mass Storage Systems, June 1994
- [23] "Storage - At the Forefront of Information Infrastructures" - Proceedings of the 14th IEEE Symposium on Mass Storage Systems, September 1995
- [24] From VLDB to VMLDB (Very MANY Large Databases), Stuart E. Madnick, - Proceedings of the 21st VLDB Conference, Zurich, Switzerland, 1995 (invited talk).
- [25] Predictions and Challenges for Database Systems in the Year 2000, Patricia G. Sellinger, IBM Almaden Research Centre, Proceedings of the 19th VLDB Conference, Dublin, Ireland, 1993.
- [26] Customisable resource management for parallel systems, Parallel Information Processing, ed. J. Keane UNICOM: Mayes K.R., Bridgland J., and Quick S. (1995).
- [27] Argante, E; Meesters, M; van der Stok, P; Willers, I, "On-line event reconstruction using a parallel in-memory database", CHEP95.
- [28] Argante, E; van der Stok, P; Willers, I, "On-line event reconstruction using a parallel in-memory database", IEEE International Conference on Engineering of Complex Computer Systems, Nov 6-10 1995.