LHC Distributed Data Management

CHEP'98, Chicago
Eva Arderiu Ribera
CERN, IT/ASD/RD45,Geneva 23,CH-1211, Switzerland
Eva.Arderiu@cern.ch

Abstract

Since 1995, the RD45 project at CERN has been working to solve the data management problems posed by the LHC experiments, where data volumes of up to 100 PetaBytes and data rates of up to 1.5 GigaBytes/second are expected. RD45 proposes the use of an ODMG compliant Object Database (ODBMS), together with a thin layer of HEP-specific code, plus a coupling to a Mass Storage System, as a solution to the object-persistency problem. This has the strong advantage of being seamlessly integrated with the object-oriented based software environment of the new experiments, offering both C++ and Java bindings. As well as satisfying the above requirements in terms of scalability, any potential solution must also function in a fully distributed, heterogeneous environment and provide efficient access to the experimental data to a world-wide physics community. Although the production phase of the LHC is still in the future, ODBMS-based solutions are already in production use by a number of other experiments (Zeus, Ceres/NA45, BaBar), and have also been used by a number of LHC-related test-beam activities.

This paper will concentrate on the data distribution aspects of the overall data management scheme, including the feasibility of using a single versus multiple collections of databases (or federations), schemes for data import and export and issues concerning data replication. We will present a set of use cases for data distribution and distributed data access that exploits the features of the Object database management system. These examples will concentrate on the use of calibration and event tag data.
Keywords: distributed object oriented databases, federated database, replication protocol

1 Introduction

Data Management has changed dramatically over the lifetime of CERN, some forty years ago. At that time, CERN used a Ferranti Mercury computer that filled a large room and had performance characteristics worse than a simple pocket calculator of today. Its clock speed was a mere 1MHz, its RAM memory capacity was 2K 20-bit words, and its storage device consisted of four magnetic drums each holding 32kx20bits. Much later on, in the LEP era, applications moved from the use of a centralised architecture, based on a "big" mainframe to a client/server model. A distributed paradigm persists to this day, but will need to evolve to cater not only for local area networks but also wide area networks and be capable of handling the quantities of data foreseen by the LHC experiments. These go beyond the petabyte range (10^{15} Bytes) and with data rates of 100MB/second to 1.5 Gigabytes/second. Although new technologies are emerging that will help us to handle these new requirements, no existing WAN network can currently offer sufficient bandwidth (bits per second) or low enough latency to allow physicists to download large quantities of data in a acceptable interval of time. It is expected that by the year 2005, the requirements of the LHC experiments will be able to be met in terms of data storage. The evolution of network capabilities and cost are, however, less certain.

In many of the current computing models of the experiments (CMS, ATLAS, BABAR) [1][2][7], it is assumed that some of the data, mainly the analysis, calibration and tag data, will be replicated and updated in remote institutes (so called **regional centres)**. Based on the client/server architecture of **Objectivity/DB,** a possible distributed scenario could be the one shown in

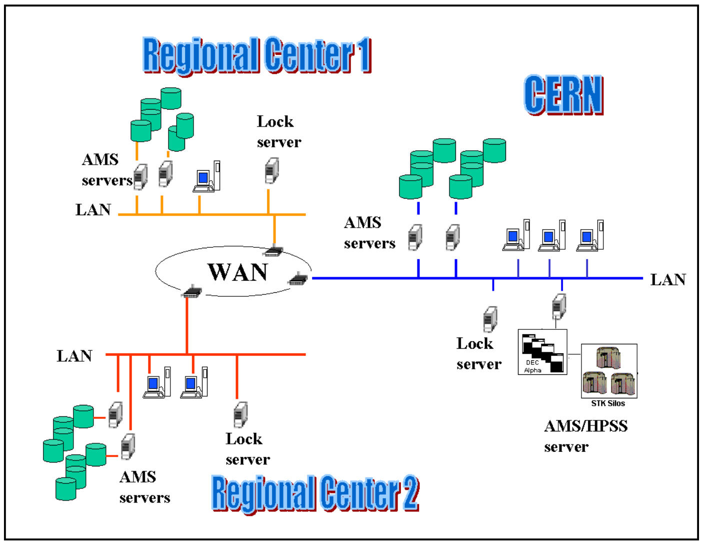

Figure 1 - Distributed Scenario for Regional Centres

figure 1. The highest logical level in the storage hierarchy is called a ***federated database (FDB)***. This FDB can be divided in several **autonomous partitions** to provide a fault tolerant environment. Each autonomous partition is formed by one or more data servers (called **AMS**), a **lock server** (which maintains consistency when concurrent access occurs), and **databases and replicas** of databases which map to files. Each database contains objects that can be accessed directly by the application.

In figure 1, we see a general view of a possible database distribution scheme. In this scenario, there is one autonomous partition residing at CERN and the others located at each regional centre. These autonomous partitions have their own lock servers and data servers (AMS) offering a complete fault tolerant environment. Should one partition become unavailable for whatever reason, the other partitions will not be affected[1] and can continue accessing data residing in databases local to their partition, or in other partitions that continue to function normally. Each autonomous partition contains databases that (typically) reside locally and/or replicas of databases that also reside in other autonomous partitions. In the case of database replicas, there is no "master" copy – each database maintains an identical and indistinguishable copy of all contained objects. The synchronization between replicas is done via the AMS servers in parallel and it is based on an **asynchronous replication protocol** with **quorum calculation**. It is possible to assign weights to the database replicas in a way that those partitions that have more than 50% of the votes are able to update replicated databases: others are only permitted to read the data.

This architecture based in one FDB and many autonomous partitions is discussed in more detail in next sections. There we study its feasibility for data distribution based on current tests and we propose alternative architectures.

2 Client/server Architectures

Based on the client/server architecture of Objectivity/DB we can adopt two distributed models: one based on a single, global federated database (FDB), as explained in the introduction, and the other based on multiple, independent FDBs.

[1] In the current version of Objectivity/DB, certain operations, such as additions or changes to database schema or the database catalogue, require that all partitions are available. This restriction will be removed in a future release of Objectivity/DB.

2.1 One Federated Database

An FDB is the highest logical level in the ODBMS hierarchy. As is shown in figure 1, we have a world wide federated database split in autonomous partitions, which communicate, via a replication protocol. This asynchronous replication protocol, together with a quorum calculation, keeps with all replicas (DB images) consistent, within a fault tolerant environment [1].

2.2 Independent Federated Databases

The second architecture proposed is based on completely independent federated databases. This means that, instead of having an FDB with multiple autonomous partitions, one per regional centre, we have separate FDBs in each regional centre, each of which is completely independent. To some extent, one can think of these independent FDBs as being "equivalent" to autonomous partitions. Instead of database replicas, there are now copies. As such, the synchronization of these copies is not handled via the DB replication protocol. Instead, an in-house procedure must be implemented. The advantage that this system has over a single federation is the lack of dependency between the different federations, which can result in increased fault tolerance, albeit at the cost of additional management overhead.

2.3 Main Differences

Each of the implementations described above has both advantages and disadvantages. Below, the most important features that differentiate the two models are given. These should be considered when deciding which choice to implement.

- **Schema Maintenance:** in a worldwide collaboration, where data is replicated to many centres, it is not easy to keep the schema synchronized. Using a single FDB, all such changes are automatically handled by the ODBMS.
- **Fault Tolerance:** if any component in the system has a failure the other ones should not be affected. Nowadays[1] with one FDB model it is not possible to change the catalogue if one of the autonomous partitions is not available. With independent FDBs, fault tolerance across regional centres is guarantied.
- **Replica Maintenance:** keeping replicas synchronized is done automatically by the replication protocol in the single FDB model, whilst in the independent FDB model this requires an in-house implementation. The **data replication protocol** from Objectivity/DB is asynchronous. This technique captures changes and stores them until the update transaction completes. At the time of transaction completion, the changes are

transmitted to the other images. If one of the receiving systems crashes or is unavailable, or there is a network failure the transaction will continue with the available ones. Inconsistent databases are resynchronized as soon as they become available. Choosing one or the other depends on the system requirements, *is it needed to have replicas consistent immediately?*

- **Object Associations:** ODBMS offer the capability to store complex object models in the database. Is it possible with one FDB model to have objects distributed over many databases. Moreover, these databases may be distributed world wide, so a user accessing one object in a certain DB can follow a link to another object that is in another DB. This feature is not possible with independent FDBs, each of which has entirely separated OIDs.
- **QoS:** another factor to consider is the quality of service (QoS) offered by the different Computer Centres which are going to support an FDB: expert manpower to handle DB failures, storage capacity, link bandwidth, etc. For example, in the one FDB model, when there is an update of a replicated DB, the updated data must be sent to all the remote sites which have the same replica too. If the network links are poor, this will slow down all the system, so it is required in such a model a global QoS respect to the network bandwidth.

Table 1 at the end of the paper summarizes all these points.

3 Relevant Availability and Performance Parameters

There are four relevant availability and performance parameters that should be considered in both architectures: number of replicas, nature of transaction, frequency of synchronization and bandwidth of the link. This section discusses these parameters and presents tests based on the single FDB model. Table 2 at the end of the paper summarizes this section.

3.1 Number of Replicas

In the single FDB model, we have one or more databases, each of which may be replicated in multiple autonomous partitions. Every update transaction that is updating data in one of the replicas synchronizes with the rest of the replicas. Typically, this means longer transaction and longer waiting times. Even though this section discusses a test with many replicas, such a scenario is not recommended in the WAN, as the bandwidth and latencies of these connections may result in unacceptably high transaction overhead.

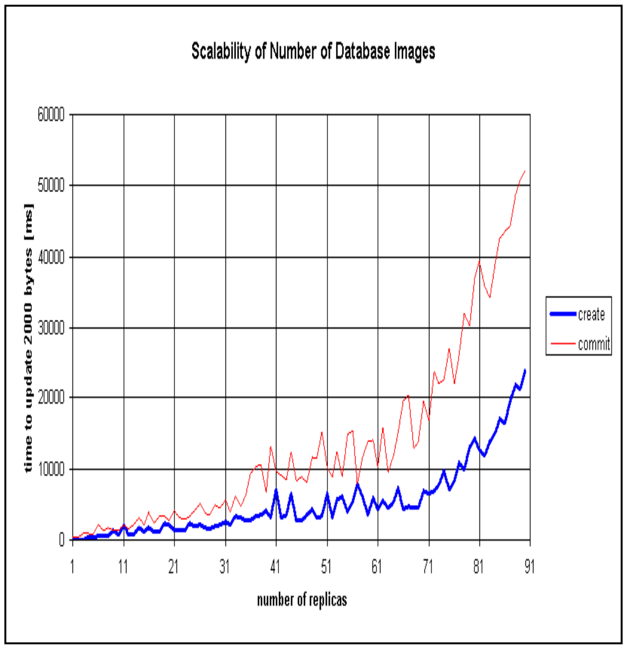

Figure 2 Replica Scalability

In certain scenarios, such as the replication of calibration or tag data, it is likely that large numbers of replicas (images) would be involved. Comparisons with HEPDB [4] suggest that some 10-15 images might be required for calibration data, but many more for tag data - perhaps as many as the number of institutes involved in an LHC collaboration. We have therefore tested replication up to 90 images - the limit arriving from the number of nodes that could conveniently be used for this purpose and not from any limitation in Objectivity/DB.

The replica scalability tests where done using some 90 inter-linked workstation-style computers running Unix. Data transfer was performed via Ethernet.

As is shown in figure 2, the time taken, both to create persistent objects and commit the corresponding transaction, increases with the number of images involved. The latter is expected - not only does the transaction not complete until the data involved has been safely written to disk on all servers, but more network traffic is involved. When an attempt to create new objects is made, the database will dynamically contact all servers involved and only permit the operation to continue if sufficient quorum is obtained. This technique, similar to that deployed in VMS clusters, ensures that database consistency is maintained.

3.2 Nature of Transaction

A **transaction** is a unit of work an application applies to a federated database. Transaction control is used to make several database requests or operations appear to all users as a single, indivisible operation. Transactions can be in update or read modes, and can perform long or short operations.

Applications do not work with Objectivity/DB objects directly; instead they work with local representations of objects which are stored in the client's cache memory, which must be retrieved from and written back to a federated database at commit time. In order to support concurrent access there is a **locking protocol** that makes sure that any user at any time accesses consistent data.

When we talk about update and about synchronization we must consider the *locking factor*, many users may access a database at the same time and lock those data that is being updated. In the ATLAS and CMS Computing technical proposals, it is assumed that some 150 physicists will be actively performing analysis at any time of day or night. Here, "active" is taken to mean the number of users who are reading or writing data to the database within a given time interval - say one hour. Given this scenario, it is important to know the type of transaction, read or update, and the amount of replicas to update per transaction.

Read transactions

Tests carried out at Caltech already have shown that concurrent reading of more than 100 users can be supported by a single Objectivity FDB without any significant performance penalty. These tests where carried out without the use of replicas. In read transactions, there is a small protocol overhead in contacting the other lock servers, but the number of replicas does not seriously affect the transaction time.

Update transactions

In the single FDB model with autonomous partitions, if one database is being updated, at the end of the transaction all its copies will be synchronized with the replication protocol. This means that the transaction time increases by a factor of the number of the replicas and the amount of data to be replicated [6]. In the independent FDB model the database synchronization is done after the transaction finished (after the commit).

To reduce locking factor and improve I/O throughput we can use concepts of parallel DBs, for example we can distribute as many data servers (AMS) as necessary, for example thousands of databases could be stored on separate data servers, handling just one user. As the locking granularity is at the level of a container, each single database could support 2^{16} parallel writers without any lock conflicts. CERN has recently installed data servers that will serve a few hundred GB of disk on multiple disk controllers, connected to fast networks, so as not to limit the I/O throughput. Clearly, an important issue will be designing a computing environment such that the individual database servers can provide sufficient bandwidth and whereby the data is efficiently clustered, as is discussed section 3.4 below.

3.3 Frequency of Synchronization

There are different ways to replicate and synchronize depending if we use the single FDB model, or the independent FDBs model:

One FDB with Partitions

- **Immediate Synchronization**: data is synchronized within the transaction using the replication protocol from Objectivity/DB. Next are the steps to create the replica. There are no replica inconsistencies, if any DB is down, it is resynchronized after it comes back to the FDB.

a) creation of the replica
<u>via network</u>: *oonewdbimage [-remoteHost…]*
<u>via tape:</u>
1- *oonewdbimage [-localHost]*
2- *oochangedb -catalogonly [new location]*
3- send replica via tape to the new location

- **"On-Demand" Synchronization**: the same as with independent FDBs.

Independent FDBs

- **Inmediate Synchronization**: not available - there is no replication protocol between independent FDBs.

- **"On-Demand" Synchronization**: the databases are synchronized by an in-house procedure. Depending on this procedure, the probability of inconsistency can be very high. Next is the procedure to create the copy:

1- *oocopydb [localhost]*
2- send file by tape or network
3- *ooattachdb [new id] [remote host] remote_boot_file*

3.4 Link Bandwidth

It is important to stress that the required network bandwidth for large databases is not yet available - it is not realistic to replicate large data volumes, e.g. in the TB range, over the networks that are typically in use in HEP today. Thus, in the short term, replication via tape is viewed as the most appropriate option for large data volumes. However, replication remains a viable solution for smallish data volumes, such as in the case of calibration and event data.

Any application requiring access to remote data has a performance penalty as the transmission of request and data are done via the network, so it is orders of magnitude slower than local access. Data replication transparently synchronizes all copies when an update occurs. All copies are thus system maintained and consistent. By providing local replicas of remote databases, local response time improves. Only updates result in network traffic, and even then, the transmission takes place only after the completion of the local transaction.

Objectivity/DB is built on a page server architecture - this means that the unit of transfer and storage is a page, the size of which may be in the range 1-64KB. Other architectures are based on object server, whereby individual objects are transferred between client and server and vice versa. By improving the clustering, i.e. the number of “interesting” objects that are co-located, we can reduce the page reads and hence the network bandwidth required and disk I/Os.

The amount of data to be copied (replicated) is a parameter directly related to the network bandwidth. The volumes of data foreseen for the LHC era, depending on the model adopted (centralized, partially or fully distributed), would require data connections between CERN and regional processing centres from a few Mbits/second to 600 Mbits/second - 2.4 Gbit/second [5]. Though we can not predict how network will evolve in coming ten years, we can already make some assumptions based on the current architecture and its scalability.

Independent of the volume of data to be replicated, databases themselves are limited in size by the system file size limits. Today, most operating systems typically support 64-bit file systems and hence the limits on file size imposed by the file system are not important – other considerations suggest that a file and hence database should not today exceed a few GB in size.

We made a test to simulate the update frequency and data volume of a calibration database, involved updating 1KB of data every 5 minutes. This was performed using a local DB that had two replicas, one located in the same LAN and the other located at Caltech (WAN access). As the figure 3 shows, the data rate was strongly correlated to the hour of the day. During peak hours, when the link was essentially saturated, a relatively low data rate of around 2Kbit/second was obtained. However, during off-peak hours, data rates of 20Kbit/second were observed. Under such conditions, remote replicas behave essentially identically to local ones. The data servers involved in these tests where HP712/60 (HP/UX 10.20), RD/6000 Power2 (Aix4.1) and Pentium Pro 200 Mhz (WindowsNT 4.0).

In figure 3 the overall throughput of the protocol stack, seen from the user, is shown. In this protocol stack we do not include the interaction with the Mass Storage Interface (HPSS): all data is disk resident.

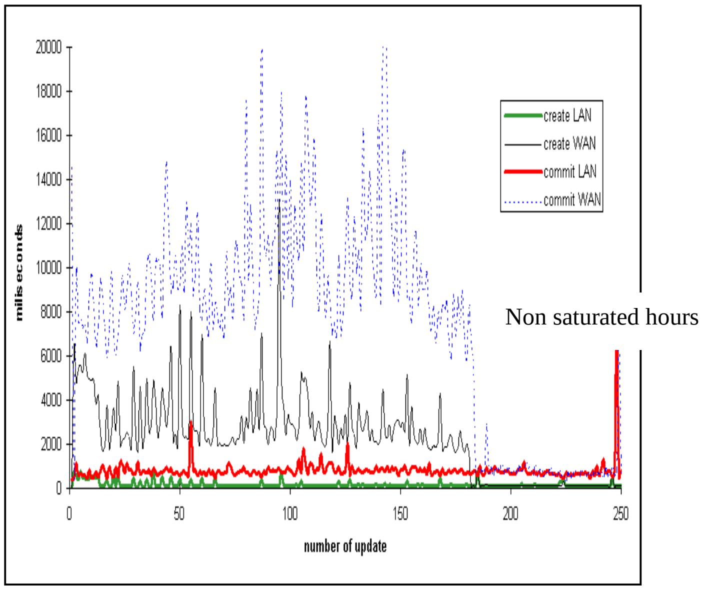

Figure 3 Comparison of update in WAN and LAN using replication protocol

4 Use Cases

Two typical use cases for replication on HEP scenarios are the event tag data and the calibration data distribution.

If dedicated networks offer the required bandwidth, the scenario based upon a single FDB would be optimal. The federated database would have as many partitions as regional centres, with the full data sample stored at CERN and replicas of the event tag data, calibration data etc. at regional centres. Direct navigation from the AOD or ESD data to the raw data would be possible. Associations could be tested for validity, probably not at the level of objects due to performance reasons, but perhaps at the level of containers, using the is_valid function. *ObjectHandle.is_valid()*. If, however, the required bandwidths do not become available, we must consider alternative architectures, such as the independent FDB model, or as discussed in section 5, a mixture of the two models described until now.

4.1 Event Tag Data

Event Tag Data contains the most commonly accessed values of the event data and thus provides a mechanism whereby event selections can be performed more efficiently.

However, unlike previous models aimed at improving the performance of the analysis stage, a link between the event tag and full event data is maintained, allowing navigation from the tag to the complete event. However, the use of multiple federated databases is not suitable for this case, as it does not permit navigation from e.g. the event summary data stored at a regional centre to the rawdata stored at CERN. This is because the object identity is unique within a federation, the objects in one database can point to objects in another database but not across federations.

Using the data replication option as in figure 1, could be used both in the local and wide area. Rather than access a remote collection, a user could access a replicated tag database. The amount of data that was replicated could be reduced further if only the tags corresponding to a specific pre-selection were made available. For certain rare and interesting channels, it would be possible to replicate the full event data too, further reducing the overhead on the wide-area network. This would not only reduce the load on any central servers, but would also minimize network traffic.

4.2 Calibration Data

The information that is typically stored in such a database includes: electronics calibrations; detector alignments; trigger/online/detector configuration; reconstruction adjustable parameters. Calibration and monitoring data, mostly produced on-line, must be available within an hour on the off-line computer in order to process data as quickly as possible. It is desirable to have this data available within a few hours on other sites for calibration studies and refits. ODBMS-based calibration systems have been developed both in BaBar and CMS. The basic functionality offered by the two systems is similar, and allows information to be retrieved based upon a "validity time".

The amount of calibration data for example in CMS will be 1 Tbyte/year.

With independent federated database configuration we loose one important aspect: in calibration data, as BaBar and CMS have implemented in their calibration libraries, users have access to the latest version or any other version of any calibration value at any moment. This feature is based on the versioning mechanism of Objectivity. If multiple federations are used, a mechanism to offer global versioning must be provided to replace this functionality.

5 Alternative Architecture

Both client/server models exposed until now have their advantages and disadvantages. A possible solution could be to mix both of them. Given the varying quality of network connections, a possible solution could be a combination of the two approaches described above. Institutes with good connectivity to CERN, e.g. regional centres and others, could be part of a single federation, with partitions at each site. Institutes with worse connections, and/or those that do not require immediate updates of the data, could use independent federations. In other words, these “satellite” sites would not be dependent on the network connection to CERN and regional centres, nor would they affect the operation of the “central” federation.

5.1 Central Fdb

The central FDB it is formed by those regional centres which need immediate update of the replicas, allow updates from any of them, and can offer a minimum QoS within them. This central FDB has just few partitions in order to reduce the number of replicas to synchronize and the transaction time.

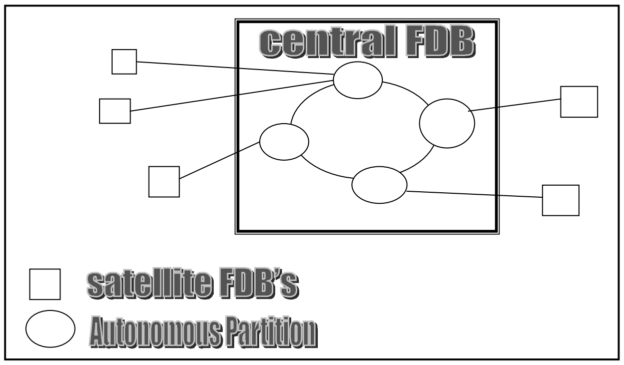

Figure 4 Alternative Architecture

5.2 Satellite Fdbs

Using standard Objectivity/DB tools, data and schema could be copied from the central federation to these satellite federations. Should new data be added at the satellite sites, this could be copied back to the central site and reattached to the main federation. As in the case with multiple independent federations, care would have to be taken to avoid conflicts in database ID – although tools now exist to attach a DB with a new identifier, changing the OIDs of all objects within that DB – and in coordinating the schema between the different federations.

6 Summary

Objectivity/DB client/server architecture offers enough flexibility to adopt different system configurations depending on experiment distribution requirements and constraints.

In this paper three possible distributed architectures have been exposed:

- one FDB with partitions: partitions located in remote institutes must offer a good QoS. Asynchronous Replication protocol should be applied to few partitions if update anytime, anywhere is required.

- independent FDBs: they offer a completely independent administration but schema and replica synchronization must be done by the DB administrator.
- merging both previous solutions. Offers a scalable distributed replica solution with a mixture of administration centres.

The distributed architecture from Objectivity/DB has been tested. A fault tolerant architecture must be offered in future releases. This is a requirement that must be met for the replication protocol to be used in production environment.

There are more tests to be done based on real use cases of the HEP scenarios, for example, updating big quantities of data and synchronizing them, modifying the schema and synchronizing it, etc..

Nowadays we are studying the combination of client/server and agent communication paradigm which could be used to combine the fact of moving the data to the client (adequate for physicists in regional centres) or moving the query to the data (adequate for some kind of are centralised access).

Acknowledgments

The tests on wide area replication where possible thanks to the help of Julian Bunn and Harvey Newman at Caltech. I would like to thank also my colleagues in ASD/RD45 for they daily support and advice.

References

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

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

[3] Adeva, B., et al. *"The L3 database system"* Elsevier Science Publishers - 0168-9002/91

[4] HEPDB, CERN Program Library Long Writeup Q180.

[5] Status Report from NT3 project. http://network.cern.ch

[6] J.Gray;P.Helland;P.O'Neil;D.Shasha: *The Dangers of Replication and a Solution*

[7] BABAR:Databases Home Page. http://www.slac.stanford.edu/BFROOT/doc/Computing/Databases/www/frames.htm

Table 1 Relevant differences between one FDB and independent FDB architectures

	One federated database	*Independent federated databases*
Schema maintenance	All users access to the common shared schema automatically	Done manually by administration centers.
Fault Tolerance	Partitions are partially autonomous (release 5.0). If one partition crashes, updates to the schema or catalogue can not be done until the AP is recovered.	FDB's are completely independent one from each other.
Replica Maintenance	The asynchronous replication protocol from Objectivity synchronises all replicas within the transaction.	There are no replicas, there are copies of DB. In-house protocol to synchronize copies.
Object associations	There can be object associations between objects from different partitions, i.e., access raw event data from event tag data.	There can not be object associations between DBs in different federations.
QoS	A QoS is required between centers involved in this type of distribution.	No QoS is required between centers, they can be off-line any time

Table 2 Relevant Availability and Performance Parameters

	One federated database	Independent federated databases
Number of replicas	They affect the duration of transaction updates. Not adequate for large number of replicas.	Replicas are attached copies. Update transactions are not affected by the number of replicas.
Transaction Nature	Update time depends on # replicas Read time profits from cache access.	Updates not affected by the number of DB “replicas”.
Frequency of Synchronization	It is handled by the ODBMS immediately on the same update transaction.	Implemented by system administrators. It Is done "on-demand".
Link Bandwidth	Small protocol overhead, depends on amount of update data and DB size	Depends on DB size.