Event Storage

Towards a Strategy on Data Persistency, Data Archive and Distribution

Abstract

The GAUDI software architecture maintains separate and distinct descriptions of the transient and persistent representations of the data objects. One of the motivations for this approach has been the requirement for a multi-technology persistency solution such that the best-adapted technology can be used. This approach has numerous advantages when testing persistency scenarios. The amount of data to be processed and the data consistency across multiple technologies was tested so far only at a small scale. In particular the data management issues like storing/retrieving large amounts of data, bookkeeping etc. In the following an attempt to complete a set of aspects is presented which will further investigation. The document is meant to initiate the discussion about the future work program and hence cannot represent a complete set of tasks to be done before the start-up of the experiment. When appropriate the position of other experiments is included.

I would like to thank the following individuals:
David Quarrie from Babar,
Vincenzo Innocente from CMS and
Fons Rademakers from ALICE,
who were going through the hassle of answering my questions.

Discussion input for the Data Management session of the

Program of Work Meeting

June 28/29

Conference Room at ALEPH pit

M. Frank

Table of Contents

1 Architectural Overview

There are many aspects of persistent data within LHCb. However, in the following I will concentrate only on the aspects of persistent event data. Still this restriction will have to address a whole variety of problem categories. The event data from the different processing stages (raw data, reconstructed data and summary data) account for roughly 0.5 PB/year. Several design decisions during the implementation of the GAUDI data processing framework support data access in a transparent way:

- Transient objects reside in a data store. An algorithm can deposit some piece of data into the transient store, and these data can be picked up later by other algorithms for further processing without knowing how they were created.
- Data stores are populated only on demand of Algorithms in order to minimise the overhead of creating unnecessary objects.
- The transient data store also serves as an intermediate buffer for any type of data conversion, in particular the conversion into persistent objects. Thus data can have one transient representation and zero or more persistent representations.
- The choice of the technology used to create the persistent representation should be largely a free parameter. It was felt unhealthy to lock into one single technology too early.
- Very early in the design phase it was felt that only one storage technology would not be enough. While a fully blown database solution was thought to be appropriate for the bulk of the experiment's data, a need was identified to also store events using a lightweight persistency mechanism, which can be used interchangeably. A possible use case is to still be able to do software development when being offline e.g. during travel, or to mail interesting events to colleagues without the necessity to export a full-blown database technology as well.

Although the chosen persistent technology has no influence on the transient world, the reverse is not applicable. The model of the data in the persistent world is heavily influenced by the organisation of the data in the transient world. The transient data store is organised tree-like, where each node of the tree may contain not only data members, but also other nodes containing further groups of data members (see Figure 1).

Any persistent technology used to store the persistent representation of the object must allow to:

- Select the correct database type containing the persistent representation of the object. The database type represents the different technologies like ROOT, Objectivity, etc.
- Locate the object on the storage medium.
- Validate the object properties according to the persistent representation.
- Handle the object's dynamic behaviour by setting up the proper function table necessary for proper calls to overloaded virtual object member functions.
- Preserve the tree-like structure of the transient data store.

This set of minimal requirements has lead to the possibility to store (event-) data using ROOT I/O, Objectivity/DB or any RDBMS supporting **O**pen **D**ata**b**ase **C**onnectivity (ODBC). Through the introduction of an extended object ID (XID) also references between different technology types are possible (see Figure 2). The XID encapsulates a universal address containing the object type, storage type, database name, container

Figure 1 The layout of a node in the transient data store

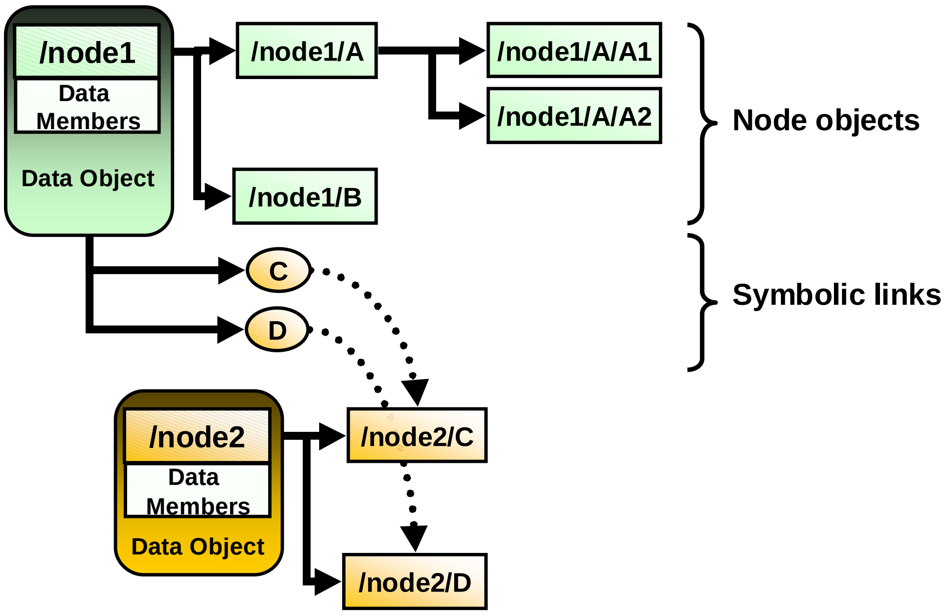
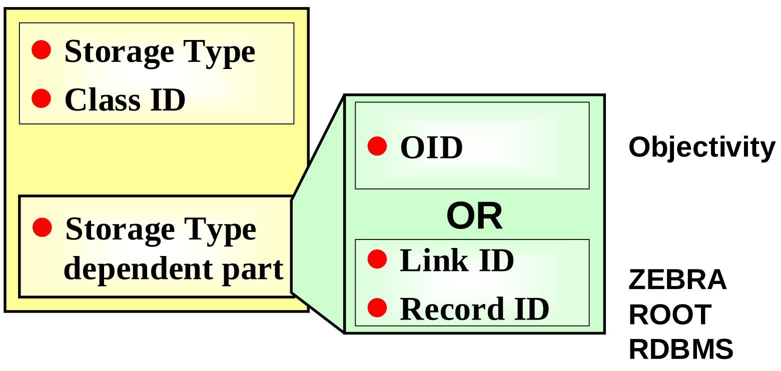


Figure 2 The information stored in an extended object id



name, object name/path in the transient store and the object identifier within the container (record ID), which allows data items to be addressed in nearly any known technology.
The separation of the data into a transient and a persistent representation has not only lead to the freedom of the choice of the persistent technology, but involved as well other constraints:

- The data must be converted from its persistent representation to its transient representation. This of course involves additional work. Currently the "schema" of the persistent representation is inside *Converters*, which are in charge of the conversion. Also a clear procedure how to break language boundaries (e.g. accessing C++ object from Java and vice versa) is unknown.
- The data management is not an integral part of the persistency mechanism. The management of existing sets of event data will have to be part of the framework. Aspects not covered by the persistency mechanism are the handling of event collections, the bookkeeping of datasets etc.

- Even though a clear separation between the persistent data and the physics algorithms was achieved, it is unclear how to manage the mass storage system responsible to decide which data are kept on disk, get purged, migrate to tape etc.

For each of these points several solutions are possible. In the following the main issues helping to draw a decision are discussed.

2 Event Data Modelling

Although called event data model, this section is not focusing on the concrete layout of classes used by the physics algorithms, but rather the meta-model used to describe the data itself. Issues concerning the data modelling include several aspects:

1. Currently the data are stored as **B**inary **L**arge **Ob**jects (BLOB).
2. The data description is currently done using C++ header files. The C++ language does not support data dictionaries allowing access to the description of object data from the class definition itself. This is a deficiency of the language.
3. The higher-level data organisation to access event data using event tags and event collections is not yet addressed.
4. Is schema evolution for event data sufficiently supported?
5. Are updates of the schema handles sufficiently supported?
6. How does the data model support the online environment?

When discussion these issues also the position of the other LHC experiments (+Babar) are included as far as the representatives responded.

2.1 BLOBs

Storing objects as BLOB has clear advantages: There is only one type of objects known to the persistent technology.

Pro:

- It is not necessary to constantly update the persistent schema which has considerable disadvantages e.g. when using Objectivity/DB.

Con:

- Low level access to individual object properties using data viewers is lost. Usually these viewers that are supplied by the database vendors are of very limited functionality anyway.

This is no issue for the any of the other experiments (at least so far).
CMS uses the explicit class description stored in the Objectivity federation.
ALICE uses ROOT, which uses the class description through the source code.

2.2 Data description

Within the LHCb data model - both persistent and transient - is embedded in the code. The C++ language does not support data dictionaries allowing access to the description of object data from the class definition itself. This deficiency inhibits:

- The automatic generation of *Converters* responsible for the creation persistent representations.

- The automatic generation of Java classes encapsulating the data of the C++ object.
- The handling of schema updates. Since at least two representations exist, a transient and a persistent representation, schema updates can occur in any of them.

A possible solution is to describe the data in a language independent way. This is a known procedure in HEP and was used in the past by SICB, Adamo and others.

Pro:

- This approach allows the generation of the corresponding header files from the description file. It would be possible to generate the class definitions for both, C++ and Java. The automatic generation of accessors could support methods object encapsulation.
- A data dictionary could be established which allows the generation of converters capable to translate between the transient and the persistent representation.
- The dictionary could be stored together with the data when the database is populated. This is advantageous because the interpretation of the stored data comes directly with the database and only very generic modules would be necessary to understand the persistent data.

Con:

- The code generation would imply the use of "yet another" pre-processor. Customised pre-processors restrict the use of the programming language because they understand only a limited number of predefined statements. In the past this was a common argument against Root's cint or Objectivity's ooddlx. The existence of off the shelf pre-processors, which can be customised in an appropriate way, is unclear.
- This approach is only sufficient to describe an object's data. It is very difficult to describe the objects dynamic behaviour by setting up the proper function table necessary to support polymorphism and inheritance.

ALICE: Root supports a reflection interface similar to the one of Java through supported by CINT.
Babar: No good idea. This is where Java would help. Also Objectivity has a reflection interface for C++ called *Active Schema.*
CMS: Uses Objectivity's pre-processor ooddlx.

2.3 Event Tags and Event Collections

Current understanding involves that events are not accessed directly. There should be an intermediate layer of accessing event through a collection of *Event Tags*. These tags should allow to pre-select events based on few basic criteria summarising the event. Said this, event tags are like N-tuples with additional support for the interpretation of the reference to the event root object. The functionality of these tags is very different from generic event data:

- The pre-selection mechanism must be suited for low I/O capabilities. Typically a set of user-defined criteria must be met in order to select an event. Event tag constructs are clear candidates to support query statements.
- Event tags will exist in may flavours:
  - Very basic event tags for the access of raw data.
  - Event tags to access a collection of events produced by a given reprocessing.
  - Event tags defined by individual users or an analysis group.

  There will not be a single definition of the event tag, but rather many definitions based on the needs of physics analyses.
- Although the possibility exists to process event tags sequentially, like it was done for the LEP experiments, a technology must be chosen to fully exploit search capabilities of the underlying technologies for enhanced queries. If enhanced query statements are needed, this must be supported as well by the storage technology e.g. by implementing an SQL interface.

Since event tags are not implemented choices corresponding to their use must be done.

ALICE: In ALICE all events are stored in several TTree's. There are several TTree's: a simulation, a raw, a reconstructed, etc. Each detector and global "algorithm" (like vertex finder, global track fitter) will have an own branch in each TTree (where appropriate). Each file contains a so-called AliRun object, which is the "root" of the session. The AliRun contains a list of all other objects available in the file (like which TTree's, etc.) and the run conditions (date/time, active detectors and algorithms, physics channels, etc, etc.). After reading the AliRun object one can access everything on the file. Once you get to the TTree's via the AliRun object you have direct access to any event in these trees.
For the physics analysis they plan to work with tag databases. A tag database will contain a TTree with physics and run information. One can foresee at least the following two analysis scenarios:
1) Loop over the tag database. If an event satisfies the selection criteria one can load the raw TTree to get access to the full event. A unique run/event pair does the matching between the tag and the event data. In addition we foresee a relational database which stores the simple relation of run/event number and filename containing the event. The relational DB is created during production and will be written by all production jobs in parallel (so good locking capabilities are needed). Optionally later the RDBMS could be copied into a simple local ROOT database, since during analysis only read access to this information is needed.
2) Loop over a tag database and create an event list (TEventList) of all events satisfying the selection criteria. Next use this event list to loop in most efficient sequence over the needed raw (or other) data.
Babar has implemented two kinds of event collections: flat collections and tree like collections themselves containing flat collections as leaves. Tree like collections have write access restrictions depending on the regime: /users/xxx, /groups/xxx and /system/xxx. The persistent classes are wrapped and not presented to the user. User code only sees the transient wrappers. Flat collections are limited to ~10^5 entries.
CMS uses named event collections in form of Objectivity containers.

2.4 Schema Changes

Schema changes currently are handled using a global class identifier. This class identifier triggers the invocation of the proper converter object, which creates the transient object accordingly. Minor changes can be implemented using the object version identifier. However:

- Will this mechanism be sufficient?
- Is it possible to supply meaningful parameters when creating "improved" transient representations of objects from "old" persistent representations?

ALICE believes the schema versioning based on the version number is sufficient for ALICE. However, they are investigating methods for fully automatic schema evolution (purely based on the RTTI information) similar to the Java serialisation.
Babar: Objectivity's schema evolution mechanism is not sufficient:

- It does not allow "old" executables to access updated data.
- They use the equivalent of a GAUDI Converter to handle schema updates. However, their persistent classes have knowledge about the predecessor. Using this Objectivity specific information allows them to chain converters, which result in the same transient object.

CMS: Vincenzo believes, that the possibilities for schema changes offered by Objectivity are sufficient to support changes to meta-data (data describing event data). For event data itself this is not so clear. Their strategy is to access data objects only through interfaces. Application code then would only see the interface without knowledge of the implementing class.

2.5 Online Environment

Current understanding suggests that the GAUDI persistency mechanism is well suited for offline oriented applications. In the online environment the GAUDI mechanism is likely to give support for the high-level software triggers. Output from the high level triggers, which should be made persistent, will have to be sent to the database server(s). The technicalities involved to store the data through a network connection will have to be investigated.

3 Data Management

So far only the application centric view of data persistence with a focus on the framework was discussed. The application simply expects the data to be accessible. However, in a world of distributed data repositories and multi-tier data storage devices (memory, disk, tape, and network) the management of the data must be addressed. Bulk data like raw data or reconstructed data are unlikely to be disk resident throughout the lifetime of the experiment and locally accessible for each physicist. Figure 3 below illustrates where the role of the framework, the database and the hierarchical storage mechanism is involved.
There are multiple solutions possible; the boundaries are not strict. Dependent on the configuration e.g. ROOT is able to cover all aspects from the presentation of the data in the framework up to the file level including remote access using the World Wide Web (a). Objectivity would support the data management of the persistent objects including the management of tertiary storage media using HPFS (b). Other database technologies typically support only data backup - the databases themselves are online and disk resident. It is unlikely that LHCb can afford to have all data online. Much more likely is a mixed solution with a certain amount of data being online and the rest residing on tertiary storage.
Within Gaudi only the framework aspects are handled. All other aspects can be formulated in the following questions corresponding to several aspects:

3.1 Data volume, storage and remote access

- Is the underlying storage/database technology suited to host the expected amount of data (0.5 PByte/year)?
- Can the database be populated with the required rates?
- The online environment is not the only user of the database. There will ~100 concurrent physics analysis tasks accessing the database. Although mostly read-only there will be multiple writers to the database e.g. while reprocessing. The database must respond to concurrent accesses with the required performance. The aggregated throughput must be sufficient.

ALICE has different needs for I/O than the other experiments in terms of performance. They use ROOT and so far are happy with it.
Babar is scared whether it will be feasible to store the data for a long time in the future unless the current limitation of 64k databases will vanish. This enhancement is promised for early next year. Using several co-existing federations used by the online, the prompt reconstruction and the analysis, they managed to match the different needs. **CMS** has so far not experienced any technical limit, which inhibits the use of Objectivity.

3.2 Data Transparency

- Does the underlying storage/database technology support multi-tier storage devices? It must be possible to connect the database to a hierarchical storage management system.

- Does the underlying storage/database technology support remote data access in a transparent way?
  Following the LHCb computing model remote data access will be essential because each tier-1 regional centre will act as a data provider. CERN will be a tier-1 regional centre serving the raw data from the experiment. All other regional centres will serve simulated data.

ALICE: Migration should be mostly automatic, i.e. when disk resources run low large data files should be migrated. This should be transparent. When accessing a migrated file, the HSM should automatically migrate the database back to disk. There are plans to optimise this interface by giving the HSM pre-stage hints during a job where we know which files are likely going to be accessed.
Babar currently is using explicit staging of bulk information from tape to disk. Thus, someone who wants to have access to the raw/rec/aod/tag information for a particular collection makes a stage request for those databases to be made available (normally only the aod/tag are disk resident) and when they are, their job will be started.
After a lot of work they managed to configure Objectivity so that it all access needs can be fulfilled. Data analysis is limited by the random read performance of the RAID disks.
Babar uses HPSS only for tape management. SLAC wrote their own disk cache management and back-end to the Objy AMS servers. They have a federation configuration system that they use to map particular groups/components/clients to particular servers so they can load-balance the servers. On each server, the disk systems are split into:
Staged comes under the control of the staging/migration/purging system, although the details of that are specific to a particular server (e.g. the Prompt Reconstruction servers don't stage databases back from tape).
Resident acts as a write-through cache: data is never purged. This is where the aod, tag or collection databases reside.
Dynamic does not get migrated. This is where the catalogue and meta-data databases reside. They change rapidly and if migrated to tape a lot of space on tape would be wasted. Backups are performed regularly.
Test is not connected to HPS at all.
CMS uses for the time being the Objectivity interface to HPSS. They will investigate the use of a customised AMS server.

3.3 Access Policy

Is it possible to restrict data access?
Since each regional centre will act as a data server accidentally accessing all raw data will cause significant load on network resources.

ALICE accesses the data through files and tapes. Any restrictions that apply to these technologies can be used.
Babar will start to allow remote access after they have implemented security enhancements in AMS. Access is restricted to selected sites.
CMS: Restrictions will definitively applied.

3.4 Data Replication

Is it possible to replicate data on remote sites if access to data is not possible via the network or improved performance/fault tolerance is required?

Babar uses data replication at the database level not using any of Objectivity's functionality for several reasons: First, several operations (database creation, catalogue updates) require **all** remote partitions to be alive (i.e. they are not quorum based operations). Second, transaction commits are synchronous and are limited by the slowest partition. Third, there's no concept of nearest neighbour for remote sites. The conglomerate has a single nearest-neighbour configuration, so if SLAC, LBL and IN2P3 have replicas, the other institutions in France might end up using the LBL replica even though the nearest one is obviously at IN2P3.
CMS assumes that they could live with mirroring using ftp i.e. the Babar approach. However, they hope that AMS will help to replicate data and more important to find the closest copy.

3.5 Re-clustering of Data

Investigations done by the CMS experiment showed that data re-clustering could lead to performance improvements. The current persistent implementation does support re-clustering only by creating partial copies of existing sets of data. No "intrinsic" re-clustering supported by the underlying database technology is possible.

ALICE: ROOT TTree's allow very easy and powerful clustering of objects in branches. The granularity of clustering can be easily tuned, for example some trees are clustered per detector and algorithm (so one can access individual detector and algorithm data), some trees are clustered on the object attribute level (allowing the access to single object attributes, e.g., used for the tag DB). So re-clustering is not an issue when using ROOT.
Babar will follow up the progress of CMS. For the time being they believe it is better to copy the interesting events from the dataset.
CMS: It is unclear if "automatic re-clustering" is feasible. Currently they are testing a prototype, which does partial event copies to a local disk on the basis of pre-selections.

3.6 Client/Server Side Processing

To improve resource load and usage it might be useful to split the data access procedures:

- I/O demanding procedures should be implemented on the server side, hence saving network resources.
- CPU demanding procedures should be implemented on the client side being a computing farm at a regional centre, an university department or even the physicists desktop.

This influences the requirements on the client-server model of the storage technology.

Babar: not used.
CMS uses at present only client-side data access although the present prototype is split in two threads: one selecting on meta-data and the other running user code. In principle the two threads could become two processes and the selector-on-meta-data be run on server-

side. Vincenzo thinks that this latter should be investigated in the context of the GRID project.

3.7 Bookkeeping

Data will be accessed according to several criteria:

- The time the data was collected at the experiment, run or fill
- According to a given version of the reprocessing
- According to physics selection criteria

ALICE: During MDC2 MySQL was used as a run catalogue database.
Babar: They extract a listing of the available collections (>>50000 now) and use an Oracle database to keep people informed of what's available. If people know exactly what collection they are working on then they can obviously bypass that. At the same time as that list is updated, a map of all the databases format that are associated with each collection for all components of the event. This information again goes into the Oracle database.
CMS has the notion of a dataset which has meta-data associated. A structure built on top to follow dataset evolution (reconstruction passes) is possible.

3.8 Processing Parameters

Storing the event data is not sufficient. To ensure reproducibility also the parameters used to produce the dataset must be accessible.
ALICE: See 2.3 (Tags and event collections).

4 The Solution Space

Some of the issues mentioned above will not be satisfied by any data base technology. However, an answer to most of these questions must be found prior to finally select an appropriate storage technology. There are essentially four basic database alternatives where experience in the HEP community is present:

- Object databases: Objectivity/DB used e.g. by Babar, CMS
  Objectivity is a client server architecture with a thin server and a thick client leaving most of the work to the application process.
- Relational databases: e.g. ORACLE
  Although is was felt during past R&D activities for the LHC experiments that relational databases cannot host event data, there are some re-considerations ongoing. This rethinking was mainly triggered by the constant success of relational databases for commercial applications (namely ORACLE) whereas the predictions for the usage of object databases were mainly falsified. Relational databases typically are based on thick-server architectures capable of data pre-selections, hence reducing the amount of data transferred. In the early '90ies the PASS project for the SSCL indicated that relational databases did not scale and later RD45 decided on the ODBMS approach - but 10 year in computer science is close to eternity.
- ROOT: used by e.g. ALICE, STAR, and PHENIX.

The ROOT I/O system is file based. Using ROOT alone however is not sufficient. All experiments using ROOT I/O to store event data have adopted a hybrid solution with a file based event data store, which usually is accessed through a relational database used for the bookkeeping.

A scenario suited for LHCb finally has to be chosen. It is clear that the cross product of all possible choices cannot be evaluated. Therefor a suitable strategy during which all necessary answers for the final decision must be worked out. One possibility to retrieve the necessary input are Data Challenges which emulate a reduced "real world" environment.

4.1 The Role of Data Challenges

According to recent experiences of the ALICE collaboration it is necessary to perform data challenges (see http://root.cern.ch/root/alimdc2_final/mdc2.htm):

- "Never believe something will 'scale', you've been there or not"
- "Individually all components work fine, when put together only then the problems show."

A data challenge does not only offer the possibility to stress the technology used to store the data, but also to identify missing components necessary to manage the data.

ALICE already finished the second Data Challenge. They expect the need for an 85-95% test-bed in 2004.
Babar did two Data Challenges: The first one focussed on a complete reconstruction chain. The second focussed on Objectivity and in particular on Prompt Reconstruction. We demonstrated scaling up to 32 nodes, which is all we had available at the time, and was 15% of the proposed production system. In hindsight, that was on the easy part of the linear slope and most of the hard work came later in achieving the design goals, which we now have with 150 nodes.
CMS is "continuously stressing all Objectivity components: Lockserver, catalogue, AMS, HSM etc in their simulation production."

4.2 A Possible LHCb Data Challenge Scenario

In particular this could involve the following:

- Choice of scenarios to be tested:
  - Simulation
  - Reconstruction
  - Data analysis.

  Each scenario has different I/O requirements. It also has to be decided whether the whole data processing chain or the database aspects should be tested. In the latter, dummy data producers and consumers can replace the simulation, reconstruction or analysis programs.
- Choice of a "good" candidate for the event data storage which will handle the object persistency.
- Identify the elements of the processing chain necessary to test the scenario and identify the missing components.

- Identify the key parameters of the scenario to be tested and configure the test-bed to find their limits.
- Exercise the system according to the parameter set to be tested.

A realistic test of the data processing chain should include at least two stages:
1. A test of the database engine with realistic data, but dummy consumers and producers. This allows retrieving the limits of the database engine itself.
2. A test of the full processing chain using software which is likely to be used in a similar form finally in the experiment.

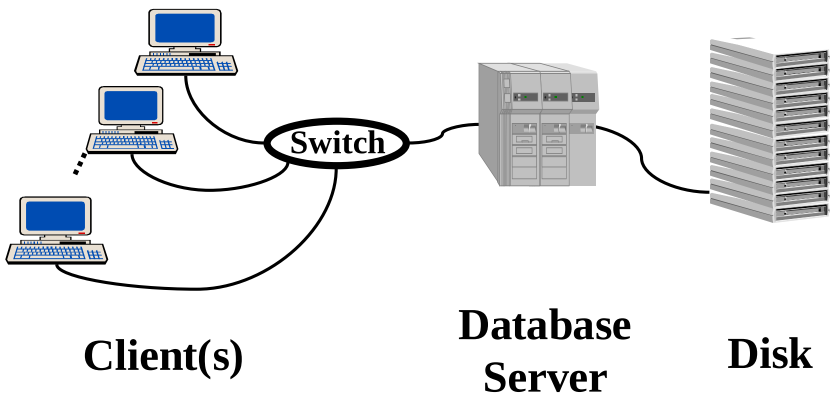
4.2.1 The First Data Challenge

The first Data Challenge must allow determining whether the chosen database engine is suitable to support the needs of the experiment in terms of
1. I/O capabilities - aggregated throughput
2. Concurrent access to the database
3. Management of resources

In order to retrieve these answers the following components must be present:
1. The test must deliver reliable results. Hence, dedicated hardware should be provided in order to minimise interference with possible other users.
2. The test database must be configured. The use of HSM has to be decided.
3. Many datasets will be produced; the bookkeeping mechanism must be operational.
4. Event access through event collections or event tags must be possible.
5. Data producers and consumers acting as dummy clients of the database must be present. These clients however must be able to simulate data access patterns similar to the final system.

Since the CPU requirements of the test-bed are limited, this should be feasible with a limited set up as shown in Figure 4.

Figure 4: A possible set-up for the first Data Challenge. Depending on the scenario to be studied, the clients can take over the role of the SFCs (Online scenario), a simulation farm or a data analysis farm. The clients should not be limited by CPU needs, but rather by I/O bandwith.



Once all necessary components are present, the following scenarios should be tested:

1. The **Online/Simulation** scenario: Feed database server(s) from one or several data producers with a typical event rate of 200 Hz (?) and an average event size of 100 Kbytes (?).
2. The **Reconstruction** scenario: Event reconstruction implies both reading raw event data and writing reconstruction objects.
3. The **Analysis** scenario: Concurrent access of many users to the same database.

Clearly the scenarios above are not as isolated as described. Reconstruction could also be connected to the online scenario in case online reconstruction is assumed to be a realistic solution.

4.2.2 The Second Data Challenge

The second Data Challenge includes the objectives of the first data challenge. Identified missing components from the first data challenge must be supplied. This data challenge should use simulation, reconstruction and sample analysis software as they will be used at the experiment's start-up. This will require a lot of CPU power. A set up dedicated to LHCb will not be possible. However, there will be a joint project for a large scale test for the high level triggers for all LHC experiments where 20% of the ATLAS needs probably correspond to a 100 % of LHCb.