Espresso – a Feasibility Study of a Scalable, Performant ODBMS

Dirk Düllmann

CERN, IT and RD45, Geneva, Switzerland

Abstract

At the request of the LHC Computing Board, the RD45 project has carried out an analysis of the risks associated with the proposed strategy for providing object persistency for the LHC experiments (namely an ODBMS such as Objectivity/DB and a MSS such as HPSS). Although these systems are being successfully used by a number of experiments, including BaBar at SLAC, one of the primary conclusions of the risk analysis is that the manpower required to develop a suitable system in-house should be estimated. This is due to the relatively small size of the ODBMS market - estimated to be worth some $100M per year - and because of a lack of a convincing commercial alternative to Objectivity/DB. We describe the goals and non-goals of the feasibility study, the main architectural features, scalability and performance characteristics and status of the prototype. We present an estimate of the manpower that would be required to design and build a full-production system and compare this with alternatives.

Keywords: ODBMS, RD45, Persistency

1 Introduction

During the last 5 years of RD45 many experiments have started to use an Object Database Management System (ODBMS) in their production systems[1, 2, 3]. It has become clear that this new technology is well matched to the specific requirements of HEP data management and offers significant advantages over traditional systems like consistent data access and scaling up to the petabyte region. The production experience gained by BaBar with Objectivity/DB have shown that this new technology poses new challenges, e.g., for system administration and performance optimisation but also that ODBMS systems can be efficiently deployed in large production systems. Even though Objectivity/DB as a commercial product is quite successful, the lack of an convincing commercial alternative became a concern especially given the long lifetimes of LHC experiments.

The Object Database market over the last years has not grown as fast as predicted even a few years ago by many experts in the database field. Several ODBMS vendors have in the meantime refocused their efforts on specific applications like management of web based documentation. Others provide their products now exclusively on the Windows platform. The number of commercial ODBMS products in the high end market is therefore still limited and as of today Objectivity/DB seems to be the only ODBMS product which has an architecture that provides the scaling and performance required for HEP data stores.

On the technical side, the RD45 evaluations and also the production experience from BaBar and CERES have pointed out several areas in which the current Objectivity/DB implementation is still lacking support for HEP specific needs. In particular the support of data distribution, support for private schema and data and the decoupling of tasks with different priority (e.g. DAQ and Analysis) currently requires to use multiple federations in parallel. Maintaining the consistency between these federations requires significant additional management effort on the application and system management side.

2 The Espresso ODBMS Prototype

The main goal of the Espresso study is to evaluate the resources needed to design and implement an alternative ODBMS as a collaborative development in the HEP (or more general physics) community. In particular we want to estimate the manpower requirements and available resources for such a development on a time scale of LHC. In addition we evaluate several new solutions for HEP specific problems, e.g. private schema or data, data replication in a WAN environment and provide a testbed for central algorithms like cache lookup and replacement strategies, I/O optimisation for wide area network and disk transfers.

It is an explicit non-goal of Espresso to attempt at this stage a complete implementation of an ODBMS product that would be usable in production as a replacement for Objectivity/DB. This would require significantly more resources than are available in the RD45 context. Espresso should rather be seen as a feasibility study for a possible joint project involving resources from many institutions in the HEP and the more general science community.

2.1 System Requirements

To guide the prototype design we have started from the preliminary requirements for a LHC data store which have been produced in RD45. These requirements are summarised in the following:

- **scalability in data volume and number of client connections**
  The architecture of the store should allow to consistently address stores at least up to the 100 PB region even under the constraint of file sizes of the order of 100GB.
- **performance of navigational access**
  The typical data access in HEP applications is navigational. The store should allow for direct lookup of any object in the store close to the I/O rate delivered by the underlying disk hardware.
- **transactional safety and crash recovery**
  The system should support concurrent read/write access of multiple client processes or threads and should maintain data consistency across the whole store. After a possible hardware or software failure it should be capable to recover to a consistent transactional state without manual intervention.
- **heterogenous access from multiple platforms and languages**
  The system should allow data access from different processor/compiler/operating system platforms and multiple languages (initially at least C++ and Java).

Given the mainly positive experience with present ODBMS systems it seemed natural to stay close to typical designs as documented in the computer science literature such as [7]. Only in areas were HEP specific requirements have been seen we have tried to come up with alternative solutions.

2.2 System Components

The prototype design has been broken down into a set of smaller component with well defined responsibilities. The main focus has been to define the main component interfaces in order to allow the replacement of a particular component implementation with a different one. For each component we have tried to identified at least one possible implementation and prototype it in C++.

The main components of the current prototype design are:

Storage Manager – This component maintains a transactional safe store for variable length data objects. The actual I/O operation is delegated to a page server instance and is only attempted if an appropriate lock can be acquired from the lock server. The storage manager maintains a client side cache which holds data within (and if possible also across) transactions. Each data object is uniquely identified by a 128 bit value (OID) which translates directly into a particular page and file for fast navigational access. The size of ESPRESSO OIDs allows to address very large stores even in the presence a limitation on maximum file size of the order of hundreds GB.

Schema Handler – The responsibility of the schema handler is to maintain a full description of the data layout within a class or structure. It provides methods to determine the name, type, position, size and data content of any object attribute and thereby allows to convert objects from one platform representation into another.

Language Binding – The language binding is responsible for a tight integration between one programming language and the data provided by the storage manager. It is responsible for retrieving data on demand using the storage manager and converting this data to the format which is expected by a particular processor/compiler combination. This conversion relies on the data layout information provided by the schema manager and should preferably follow the ODMG standard[6].

Data Server – The data server component is responsible for storing and retrieving one or more data pages between disk and storage manager. For local I/O it directly calls the host operating system I/O functions for the actual work. In case of remote access it communicates via a network with a remote daemon process which provides access to remote disk storage.

Lock Server – The lock server maintains a central resource lock table which is used by all storage manager instances to coordinate concurrent access to disk resident data.

2.3 Storage Hierarchy

An ESPRESSO store (or federation) is hierarchically structured into smaller storage units. The new concept of a *Domain* introduces a notion for a set of tightly related files (files with many cross file relationships). These domains relate closely to object model domains and allow to efficiently move data between different federations for distribution and backup purposes. Since the catalogue for different domains is maintained in different files, this design is expected to lead to better concurrency behaviour than a single catalogue for the whole federation. Each Domain can contain up to 64k *Files* each of them potentially on a different host. Each file may consist up to $2 \cdot 10^9$ *Data Pages*. On each page up to 64k *Object Slots* may be addressed. This storage hierarchy is directly reflected by the internal structure of Object Identifiers (OIDs) used by ESPRESSO to represent references to persistent objects.

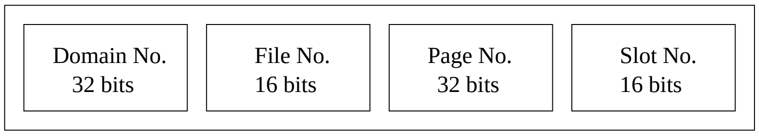

Figure 1: Espresso Object Identifier

In the catalogue we foresee the possibility to mark entire domains or single files as read-only in order to minimise the number of locks requests in the system. Since a large fraction of HEP data will not be modified after an initial write operation, this is expected to significantly increase the scalability of the system for very many clients.

3 Status of the Prototype Implementation

A prototype of the ESPRESSO store has been designed and implemented during the last 10 month as a part time activity by a group 3 developers. In order to achieve this goal we made use of more recent features of the C++ language namely exceptions, name-spaces and the standard C++ library as defined by the ANSI/ISO standard. Especially the use of STL for all containers and search structures has significantly sped up the development and led to a more homogeneous coding style. For the implementation of the networking between client, lock server and data server we have profited from a portable TCP encapsulation provided as part of the ObjectSpace class library. The main development platform is Linux with the GNU C++ 2.95.x compiler, but major parts of the system have also been compiled and tested under Solaris (CC 5.0) and Windows (GNU C++ 2.95.x).

To check the performance of ESPRESSO some first benchmarks have been done which show expected scaling behaviour and prove that even the current prototype with very few optimisation can perform close to the disk transfer limit. More detailed review of the results refer to [1].

During the next month we plan to extend the scalability tests, to complete the heterogeneity support for at least one additional platform and to provide a complete HepODBMS implementation.

4 Conclusion

A prototype of a scalable and performant ODBMS has been designed as part of the RD45 risk analysis. A set of components and component interfaces have been identified and successfully implemented with very limited manpower resources. Only very few HEP specific additions are required to offer significant improvement over todays commercial implementations. We estimate that a complete implementation assuming manpower resources of the order of 15 part time developers over a period of three year would be sufficient to provide a production quality implementation.

Given the broad interest in the availability of an alternative ODBMS implementation not only in HEP but also other research areas like plasmaphysics, astrophysics it seems feasible to complete such an implementation in the context of a new collaborative development project on a time scale suitable for LHC.

References

1 `http://wwwinfo.cern.ch/asd/rd45/index.html`

2 J. Shiers, “Status Report of the RD45 project”, CHEP 2000, Padova, Winter 2000.

3 D. Quarrie, “Operational Experience with the BaBar Database”, CHEP 2000, Padova, Winter 2000.

4 `http://wwwinfo.cern.ch/asd/lhc++/HepODBMS/user-guide/ho.html`

5 `http://wwwinfo.cern.ch/asd/lhc++/htlguide/htl.html`

6 R.G.G. Cattel (editor), “The Object Database Standard ODMG 2.0”, Morgan Kaufmann, San Francisco 1997.

7 J. Gray, A. Reuter, “Transaction Processing Concepts and Techniques”, Morgan Kaufmann, San Francisco 1993.