OBK – An Online High Energy Physics' Meta-Data Repository

I.Alexandrov[1], A.Amorim[2], E.Badescu[3], M.Barczyk[4], D.Burckhart-Chromek[4], M.Caprini[3], M.Dobson[4], J.Flammer[4], R.Hart[5], R.Jones[4], A.Kazarov[4,8], S.Kolos[4,8], V.Kotov[1], D.Liko[4], L.Lucio[2,4,7], L.Mapelli[4], M.Mineev[1], L.Moneta[6], I.Papadopoulos[4], M.Nassiakou[4,9], N.Parrington[7], L.Pedro[2], A.Ribeiro[2], Yu.Ryabov[8], D.Schweiger[4], I.Soloviev[4,8], H.Wolters[2]

1) Joint Institute for Nuclear Research, Dubna, Russia
2) FCUL (Science University of Lisbon), Lisbon, Portugal
3) Institute of Atomic Physics, Bucharest, Romania
4) European Organization for Nuclear Research (CERN), Geneva, Switzerland
5) National Institute for Nuclear Physics and High Energy Physics (NIKHEF), Amsterdam, Netherlands
6) Physics Section, University of Geneva, Geneva, Switzerland
7) Sunderland University, Sunderland, England
8) Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia
9) NTUA, National Technical University of Athens, Athens Greece

Abstract

ATLAS will be one of the four detectors for the LHC (Large Hadron Collider) particle accelerator currently being built at CERN, Geneva. The project is expected to start production in 2006 and during its lifetime (15-20 years) to generate roughly one petabyte per year of particle physics' data. This vast amount of information will require several meta-data repositories which will ease the manipulation and understanding of physics' data by the final users (physicists doing analysis). Metadata repositories and tools at ATLAS may address such problems as the logical organization of the physics data according to data taking sessions, errors and faults during data gathering, data quality or terciary storage meta-information.

The OBK (Online Book-Keeper) is a component of ATLAS' Online Software - the system which provides configuration, control and monitoring services to the DAQ (Data AQquisition system). In this paper we will explain the role of the OBK as one of the main collectors and managers of meta-data produced online, how that data is stored and the interfaces that are provided to access it - merging the physics data with the collected metadata will play an essential role for future analysis and interpretion of the physics events observed at ATLAS. We also provide an historical background to the OBK by analysing the several prototypes implemented in the context of our software development process and the results and experience obtained with the various DBMS technologies used.

Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002

1. Introduction

HEP (High Energy Physics) has traditionally been a heavy user of database techniques and products, usually at significantly higher levels than the typical market demand. This is due to the enormous amount of data samples physicists need to collect and store in order to be able to produce statistics which are accurate enough to validate their theories.

ATLAS is one of the four detectors being built for the LHC particle accelerator at CERN to be completed in 2006. After being accelerated, particles will collide inside the detector. From observing the results of those collisions physicists expect to be able to expand our knowledge on the basic contituents of matter.

Physics data ready for analysis is composed of what physicists call events – each event describes the interaction of particles and their final state products. The process of producing and storing interesting events is however a very complex problem: in the detector the real events occur at rates that require extremely sophisticated hardware and software to filter out only a fraction ($1/10^7$), which at the ATLAS may still mean 100 interesting events of 1 megabyte each, per second. Given that ATLAS is expected to run for 15-20 years, the

generated databases may go up to tens of petabytes [SHI98].

However, the data produced is not limited to events. Such large machines (the LHC and ATLAS) are composed of many sub-systems which have their role in data-taking. Since all these sub-systems are highly configurable and/or may change over time, all the parameters and running conditions will also have to be stored for later contextualization of the collected events. These additional databases may include for example software and hardware configurations of the computing infrastructure, alignment of the parts of the detector or environmental conditions like pressure, temperature and other information that need to be taken into consideration to achieve precise physics measurements.

In order to make the massive amounts of data contained in the above mentioned databases accessible and understandable by the physicists doing analysis, metadata (data about data) is useful and necessary. Although it is difficult to define the boundary between what is data and what is metadata, in this paper we take on the following definition: metadata is "*data which makes data more accessible to the user*" [BAR97]. In this sense several systems within ATLAS may be considered as metadata repositories. The OBK in particular deals with online cataloging of physics' information.

How these metadata databases are organized and accessed, how they will interact with the main databases or what technologies will they employ is still unclear, given that the subsystems are still being built and will only be ready for production in 2006. This paper tries to address some of these questions from the viewpoint of the OBK. References to other metadata repositories in HEP are also provided in order to better put the problem into perspective.

1.1 Database technology background

As mentioned before, database technology has always been a hot topic in the HEP experiment domain. Until recently however, either because of market solutions' limits of performance and scalability or because of physicists' preference for custom solutions, the HEP world and the database community often did not collaborate in finding solutions to their common problems.

Commercial databases made their appearance in HEP experiments during the 1980s for the handling of book-keeping data (metadata). The management of physics' data itself was left to specialized solutions. In the 1990s however, given the appearance of more flexible data models (Object Oriented) and the increasing performance and scalability of commercial DBMSs, the HEP community showed interest in the available technology.

In the context of the future LHC experiments – including ATLAS – the RD45 project at CERN set out to test commercial DBMSs for HEP requirements [MAL97]. The Object Oriented design model was of particular interest, given that it is a closer match for particle physics' needs in term of data model than previous relational approaches. As a result of the RD45 project, the Objectivity/DB DBMS was chosen as a vehicle for study of LHC long term data. As will be explained in this paper, the OBK also followed this trend in a first prototype iteration. Note that specific solutions are still very present in the current ATLAS database panorama, along with market solutions other than Objectivity/DB.

Currently, investigations by CERN/IT group [NOW01] point to Object/Relational DBMSs as the future of databases in HEP. Oracle 9i attracted a lot of attention for its rich data model as an implementation of SQL:1999 and also for its VLDB oriented features.

2. The Online Software from the OBK's viewpoint

To understand what kind of metadata the OBK stores and the technology context the system fits in, this section of the paper provides an overview of the Online Software, focusing on the components the OBK interacts with most.

2.1 Architecture of the Online Software

The OBK is a software component of the Online Software, which in turn is a sub-system of the TDAQ (Trigger-DAQ) – the set of software and hardware systems responsible for acquiring, filtering and moving event data from the detector to mass storage, in order to make it available for subsequent analysis.

The role of the Online Software is to provide to other TDAQ systems configuration, control and monitoring services. It excludes however the processing and transportation of physics' data [ATL00]. More specifically, the Online Software provides such services as uniformly assigning commands to other systems so that they can change state coherently, interfaces to databases which hold parameters for the configuration of the whole TDAQ, inter process communication facilities, monitoring (including sampling and data quality checks) and book-keeping services. A range of ancilliary services such as graphical user interfaces are also included.

As can be understood from the above explanation, the Online Software is a generic framework designed to be able to supervise many distinct data taking configurations. Since the ATLAS will be a very large machine whose components must work concurrently, the Online Software is also reponsible for managing that concurrency in terms of software. A subset of the detector and TDAQ hardware/software able to acquire data independently is called a *partition.*

The architectural configuration of the Online software can be observed in figure 1. The various components are bunched in groups called *super components*, which include *Run Control*, *Messaging*, *Monitoring*, *Ancilliary*

and *Databases*. In order to interact with the TDAQ systems the Online Software supervises, a custumizable programatical interface called *Local Controller* is provided to each of those systems.

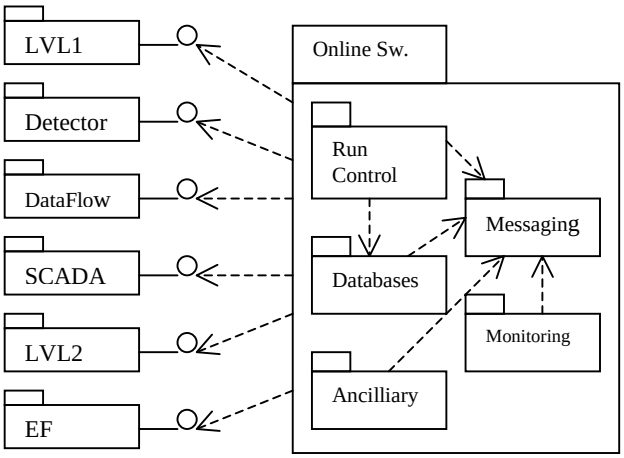

Figure 1 – Simplified UML package diagram of the Online Software

2.2 Book-keeping data sources

The OBK is considered to be part of the Databases super-component together with the Configuration Databases – although the roles of the two components are quite distinct. A high level description of the purpose of the OBK inside the Online Software is provided in [ATL00] and goes as follows: the OBK "*archives information about the data recorded to permanent storage by the DAQ system. It records the information to be later used during data analysis on a per-run basis and provides a number of interfaces for retrieving and updating the information*". In particle physicists' jargon, a *run* corresponds to a data-taking period with a certain parameterization of the systems involved. The OBK can be defined as a run cataloger.

A significative part of the non-physics information circulates inside the TDAQ by means of the Online Software's Messaging super-component, which includes the IS (Information System) and the MRS (Message Reporting System). Both the IS and the MRS use as backbone the IPC (Inter Process Communication) package, a CORBA implementation [AMO97]. The MRS is aimed at providing a facility which allows all software components in the ATLAS DAQ system and related processes to report error messages to other components of the distributed TDAQ systesm. On the other hand, the IS provides a mean of inter-application information exchange in the distributed environment. The structure of the IS and MRS messages is highly flexible and parameterizable.

Other non-physics data sources which are interesting from the OBK's point of view are the Configuration Databases. Since this component manages configuration data for all of the TDAQ, some of that data will be relevant for correctly cataloging runs. The configuration databases component relies on OKS (Object Kernel Support) [JON97] for database schema definition an data persistency. OKS is an Online Software homegrown solution and acts as an object oriented in-memory persistent object manager.

IS, MRS and Configuration Databases make data available through C++ APIs which the OBK needs to use. Figure 2 depicts the interaction of the OBK with these Online Software components.

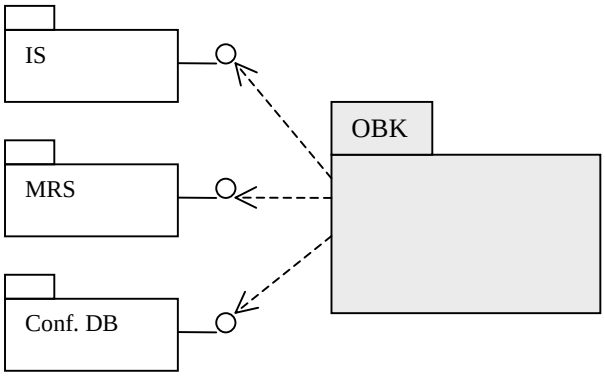

Figure 2 – Simplified UML package diagram of the OBK

3. The Online Book-keeper

3.1 Development approach

Work on the OBK component started in 1996, within the context of the DAQ/EF-1 [MOR97] project which aimed at building a prototype for a full vertical slice of the DAQ system. The implementation of this prototype had also as objectives the study of technological and architectural options, along with software development methodologies and environments.

The Online Software adopted a software development process with the following phases: problem statement; analysis; design; implementation; testing; maintenance. Work on the OBK followed this process loosely, applying the described phases to each of the three implemented prototypes.

Generically, the process used to develop the OBK can be described as prototypical spiral, in which each implementation follows the cascade approach while building on reviewed requirements and previous experience. This sort of approach seems appropriate to a long term project in which the requirements for the component evolve as the full system grows. To be noted that maintenance plays as very important role in everyday life at the Online Software, as the software is frequently

required to supervise simulated and real data acquisition tests.

Another output of the development process other than the code is the documentation. For the OBK, documents such as the Implementation Notes, the User Manual and Test Reports (according to a predefined Test Plan) are produced at each prototype iteration. Some of them will be refered to in this paper as they include interesting results.

The Online Software makes use of a number of tools to support the development. For example, CVS (www.cvshome.org) is used as the code repository, SRT (http://atddoc.cern.ch/Atlas/DaqSoft/sde) and CMT (http://www.lal.in2p3.fr/SI/CMT/CMT.htm) – both CERN homegrown solutions – for platform dependency and release management, Rational Rose and Together for high level design, Perl and Unix shells for scripting, Framemaker for documentation generation, etc. This, added to a multiple-language programming environment, makes the software developing environment for the OBK quite heterogeneous and advanced.

3.2 High level requirements and generic architecture

In section 2.2 a very high level description of the requirements for the OBK has already been provided. This paragraph extracted from the technical proposal (latest publication in [ATL00]) was the departure point for the work on the OBK.

The starting round of formal requirements collection was performed in 1997 [AMO97]. The resulting document provided a first refinement of what the OBK should do. A brief summary of that document is as follows:

- The OBK shall record information on a per-run basis. This information should include: run configuration; trigger state and configuration details; recording device details; run dates (start/stop date and time); accelerator beam information; data quality; error statistics; structured user comments;
- The OBK shall provide access to the archived information for DAQ and detector groups via a number of interfaces which may include a general user interface and APIs;
- The OBK shall allow the data taking period coordinator to modify the information stored for a particular run;
- The OBK will require the retrieval of information from the following DAQ components: MRS, Configuration Databases and Accelerator;
- The OBK shall be accessible during and outside DAQ data taking sessions.

From 1997 to today, the Online Software (at the time DAQ/EF-1 Backend) evolved in terms of design, technology, performance and scalability. Being an Online Software component, the initial requirements posed on the OBK also evolved. A formal review on the OBK requirements is in progress at the time that this paper is being written – this review will make it possible to incorporate requirements that came up during the three prototype iterations and also include new desired features.

Although the requirements revision is still taking place, it is already possible to say that the new set of requirements is mainly a constrained version of the first document. The main additions can be resumed in the following points:

- Some of the data required to be stored in the first document is now removed. However, the definition of data quality is enhanced and the OBK is also required to store data samples (histograms) for offline data quality checks;
- The fashion in which the OBK acquires data is better specified: the tool should be able to gather data either by collecting TDAQ circulating messages which are interesting or by requesting their production explicitly;
- External frameworks and tools for which the data stored by the OBK is interesting are named. This means that interfaces with those frameworks or tools should be investigated;
- The need for the implementation of a web-based interface to the stored data is explicitly mentioned.

From these requirements a simple generic architecture for the OBK can be devised. A graphical representation of this architecture can be found in figure 3.

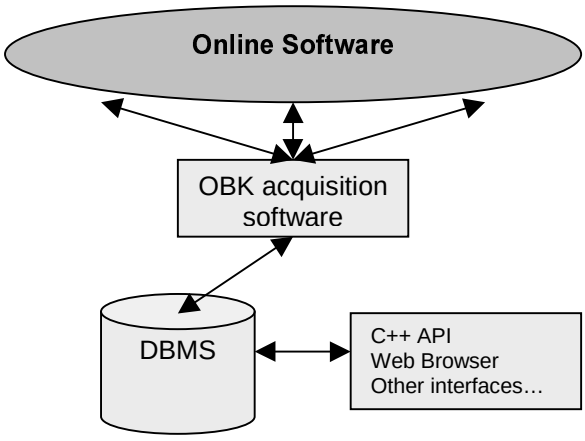

Figure 3 – Generic OBK architecture

The architecture of the OBK consists then of an application which subscribes to various data sources and stores into a DBMS online relevant data for run cataloging. It also makes available several interfaces to retreive that data offline.

3.3 Implemented prototypes

During the 4 years of existence of the OBK project three distinct prototypes were implemented. All of them are coded using C++ but different DBMS technologies: Objectivity/DB (www.objectivity.com) , OKS and MySQL (www.mysql.com). The advantages behind this multi-technology approach are to to gain technological expertise about different DBMS technologies and in general to be able to make a solid recommendation for a technological and design solution for a production book-keeper tool.

All the OBK prototypes are available in the Online Software's current release (0.0.16) – the user can choose which one to use by setting an environment variable.

3.3.1 Objectivity based

As was mentioned in 1.1, Objectivity/DB was selected by the LHC experiments study the management of long-term data. Naturally, the first OBK prototype to be implemented was based on this DBMS.

Being a pure Object Oriented DBMS, Objectivity/DB provided the possibility to explore the persistent Object-Oriented (OO) data model. Despite its power and flexibility, this data model has less acceptance than the Relational model, mainly because of the state of maturity and ease of use of the latter. The reason why particle physicists invested strongly in OO persistency in the 1990s resides in the fact that this data model is very flexible and can describe complex particle physics information better than the relational one. An example of strong usage of Objectivity in HEP is the BABAR experiment [GAP01] taking data in the United States.

This first OBK prototype was designed to take full advantage of the data model proposed by Objectivity/DB, which is organized into federations, databases, containers and objects. The mapping of the collected data was quite natural and intuitive.

Some highlights on the experience with Objectivity are the following:

- Objectivity/DB's schema definition mechanism (ODMG compliant) is very powerful and reduces the difference between transient and persistent objects. This enables the programmer to set a mindframe on a high-level language paradigm – OO – throughout all the development process;

- Objectivity/DB databases at CERN are concentrated in a small number of machines with mass storage connections (CASTOR) and running AMS (Advanced Multithreaded Server) for remote requests. The Objectivity OBK prototype made use of these services during it's deployment;

- Being a commercial tool, Objectivity/DB provides a large amount of features, including graphical browsing utilities, transaction and recovery management, etc. It has however the disadvantages of not running on all the platforms supported by the Online Software and also requiring a commercial licence, which raised problems within an Open Source developing environment such as the Online Software.

The Objectivity based OBK prototype was succesfully used for the 2000 test-beam, a full scale test for the TDAQ involving a functional subdetector of the ATLAS. A web-based browser has also been developed as part of the prototype.

3.3.2 OKS based

The second OBK prototype used the OKS DBMS for persistency. OKS is an in-memory persistent object manager implemented to satisfy the specific needs of the ATLAS TDAQ in terms of configuration databases. The strength of the OKS lies in being lightweight and oriented to clients which pose strong efficiency and real-time requirements. The data model adopted by the OKS is also Object Oriented, although not as sophisticated as Objectivity/DB's. At the moment, the OKS is the adopted solution to manage ATLAS TDAQ configuration databases. Data files (containing objects) are stored as XML files in the filesystem (local, AFS or NFS). Access to the data is done by explicitly reading the data files, not through a centralized server.

The motivation to use OKS for OBK persistency came from the fact that the commercial licence required by Objectivity/DB posed problems when it was necessary to run the OBK outside CERN (where the licenses are not valid). Another reason for choosing OKS is that Objectivity/DB ships by default with too many features that the OBK does not use. All this extra functionality creates an overhead for the OBK and for the Online Software in general. Since the OKS package is already a part of the Online Software, compilation, linking and shipping of the code becomes straightforward.

Concerning the technical aspects of the implementation, the mapping of the database schema previously defined for Objectivity/DB into OKS was relatively simple, given that both DBMSs use the OO data model. The mapping of the Objectivity/DB concepts of Federation, Database and Container on which the first prototype relied posed however some challenges, given that the OKS only has two levels in its data model

hierarchy: data file and object. Also, since concurrency control is not implemented for OKS, that responsibility was passed onto the OBK's code.

This second implementation includes some significative improvements on the first one: a more functional web browser, a C++ API, dealing with data quality and more object-oriented design.

When used in the 2001 test-beam the component behaved well, acquiring several megabytes of data which were stored on the machine's local filesystem and later on AFS. The scattering of data across filesystems was however a problem that soon arose.

3.3.3 MySQL based

Finally, the most recent OBK prototype uses the MySQL DBMS for data storage. MySQL is an Open Source product known worldwide for its speed and ease of use. Inside HEP it is also extensively used for metadata management solutions, as will be pointed out in section 5 of this paper. MySQL implements the Relational data model and access to its data is via a centralized server which processes SQL queries.

The decision to implement a MySQL based prototype was triggered not only by power of the underlying SQL engine, but also by the desire to try a relational approach while dealing with the OBK data.

Technically, the mapping of the predefined OO OBK database schema into the relational model required substantial redesigning, as the two models are essentially different. From our point of view the relational implementation of the schema is less elegant than the previous approaches.

The code itself is however less voluminous and complicated than previous approaches, given that the C/C++ API provided by MySQL enables the execution of high-level SQL queries instead of forcing the direct manipulation of persistent/transient objects.

The MySQL OBK prototype is work in progress. It is supposed to encompass all the functionality of the previous prototypes and also to address the newly collected requirements (see section 3.2). At the moment that this paper is being written the acquisition and web browsing facilities are already available and integrated with the rest of the Online Software.

In principle the MySQL OBK prototype will be used for this year's (2002) test-beam.

3.3.4 Access to the stored data

The OBK makes available several data retrieving utilities and interfaces: command line utilities, a C++ API and a web browser.

The command line utilility is an executable binary which the user can pass several parameters in order to choose what OBK data to retrieve (dump to the screen). Although not very sophisticated, this utility can be used for example in debug situations or when not many resources are available.

The C++ API consists of a shared library the user can link with his/her C++ applications to have access to the OBK data as C++ objects. The library makes heavy use of STL to compose the query return structures.

The web-based OBK browser makes available a graphical version of the data through a regular HTTP client (Netscape, Iexplorer, etc.). The mechanism relies on PHP scripts to access the database along with the Apache HTTP server to make the information available in the web. We currently have a machine deployed to provide this service. In figure 4 a snapshot of the browser can be observed.

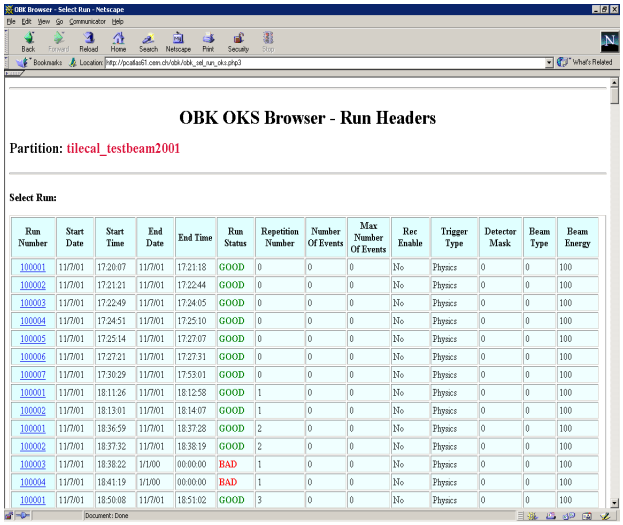

Figure 4 – Snapshot of the OBK (OKS implementation) web browser

3.4 Design and implementation issues

In this section we comment on some of the design and implementation issues we found to be more relevant while developing the three prototypes. Although some of the remarks in the following list may seem to ovelap with section 3.3, we try to pose them in a more generic fashion:

- **Database schema:** as mentioned before, the OO data model was found to be quite flexible for mapping the OBK data. The resulting database shemas (for Objectivity/DB and OKS) are very natural since each type of data to be stored is modelled into a class. That relates to other classes in the schema through inheritance or composition. The Relational data model proved to be less easy to use: data types such as collections had in fact to be treated as XML strings, given that the data model does not support the

concept. The relational OBK database schema is less natural than the OO one, given that classes had to be fitted into tables and also that optimizations were performed in order to facilitate access to the data;

- **Code:** we found a very noticeable tradeof between the elegance of code achieved with the OO data model and the reduced amount/complexity of the code in the relational prototype;

- **Evolution of the design:** while within the context of an online system, a strong requirement for the OBK is light weight in terms of processing and primary memory usage. To fulfill this purpose we tried to evolute the design in terms of minimizing accesses to the database while keeping in primary memory the least possible amount of data. In terms of data retrieval we found that using caches for frequently accessed run parameters reduced drastically read times;

- **Use of XML:** in the MySQL prototype, XML was used to cope with the difficulty of storing collection types, not available in the Relational data model implemented by that DBMS. Despite not being too elegant, this technique seems to simplify storage of data during acquisition and could even be used in conjunction with the previous OO prototypes to avoid the creation of too many simple data objects.

3.5 Performance and scalability issues

The Objectivity/DB and OKS OBK prototypes were used in test-beams and Online Software scalability tests, as part of the Online Software. Although some bugs were found, no major performance or scalability problems were identified. However, these tests envolved only a small fraction of the final system the production OBK will have to handle. To be able to quantify the capabilities of the OBK we also performed tests in a controlled environment. Some of the results are described in the following lines.

Figure 4 shows the storage time of an important MRS message for the three prototypes. The message in question indicates the start of a new run and triggers heavy processing within the OBK. As can be seen, the Objectivity/DB prototype is the slowest while the MySQL one is the fastest. For the Objectivity/DB and OKS prototypes a non-desirable growth in storage time can also be observed. We attribute these differences mainly to the evolution of the design from prototype to prototype. We think that better tuning of the OO OBK solutions would place them closer to the MySQL one in terms of performance. More about these tests can be found in [ALE01].

Scalability problems can exist in two different forms for the OBK: in terms of data storage or in terms of data sources. In what concerns data storage we are confident that no major problems will arise, given that metadata repositories are by definition orders of magnitures under main databases in terms of occupied space. In any case, if secondary storage is not enough, CERN makes available a terciary storage service (CASTOR).
In terms of data sources, the OBK (OKS implementation) was tested to understand what happens if interesting data is produced at a rate that the OBK cannot cope with. The tests are described in [LUC02] and involved starting a large number of data sources simultaneously and making the OBK subscribe to them. When overflowed, it was observed that the OBK may cause delays to other Online Software components. Although the simulated situation was much above what may be required by a production system, the results are indicative that the storage time for the OBK should be kept as low as possible.

All the above mentioned tests were performed using Pentium III client and server machines running Linux.

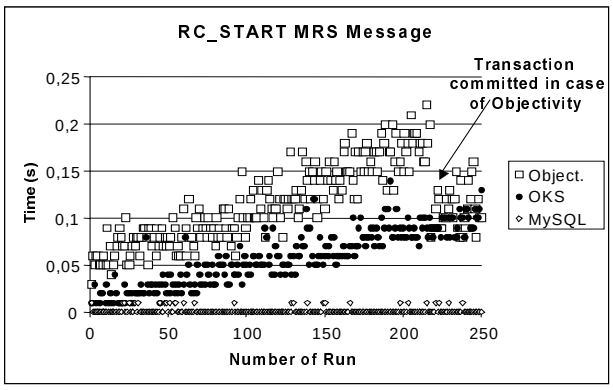

Figure 5 – Comparative performance of the three prototypes

4. Conclusions and future work

The OBK is work in progress. From the start of the project in 1996 until now effort has been put into understanding the problem of book-keeping for ATLAS Online Software. Given the long duration of the project, the used software developing process emphasized the study of alternative designs and technologies by implementing three prototypes based on different DBMS technologies: Objectivity/DB, OKS and MySQL. These alternative implementations also enabled us to easily incorporate new requirements for the component as they appeared in the context of the evolution of the TDAQ.

At the time that this paper is being written we think we are already in a good position to recommend the technology and the design for a production book-keeper. However, we must also take into consideration that until 2006 ATLAS' Online Software may still change and that technological advances in computing happen at an extremely fast pace. An implementation of the OBK using Oracle 9i is not excluded.

Also to be mentioned that the most important relevant quality of metadata is that it should make data more accessible to the user. Given that the ATLAS TDAQ systems are now starting to move torwards integration, in the close future we will have to concentrate on how the OBK data will integrate with the main data repositories, i.e., how to make the OBK data useful.

5. Related work

In this section, some examples of what is being currently done in the area of metadata repositories in HEP are provided. The set of mentioned systems is interesting from the point of view of the OBK and does not by any means try to represent the general state of affairs in the area.

Inside ATLAS two systems which are very close to the OBK are the *Liquid Argon*[1] *book-keeper* [ALB01] and the *Tilecal*[1] *book-keeper* [SOL01]. Both of them store run catalog data and use as underlying technologies PHP and MySQL.

Another interesting metadata repository being used in ATLAS, this time for cataloging and replication of distributed data, is the *Magda* project [WEN02]. Magda is currently being used by the offline ATLAS Data Challenges for keeping track of event data generated in a simulation context. Technologies used include MySQL, Perl, Java and C++.

Finally, although the *Spitfire* project of the DataGrid Data Management Work Package [HOS01] is not a metadata repository on its own, it aims at providing middleware for the integration of heterogenous SQL based metadata repositories. Spitfire's purpose is to act as an intermediary between clients of metadata databases in very large distributed systems. Underlying technologies include (among others) Java, XML and MySQL as the default database backend.

References

[ALB01] Solveig Albrand and Jerome Fulachier, "Bookkeeping Database Search Interfaces", URL: http://larbookkeeping.in2p3.fr/

[ALE01] Igor Alexandrov et al., "Experience using different DBMSs in prototyping a Book-keeper for ATLAS' DAQ software", *Proceedings of CHEP 2001*, Beijing, China, 2001, pp 248-251.

[AMO97] Antonio Amorim and Helmut Wolters, "Requirements for the Run Book-keeper system for the ATLAS DAQ prototype –1", URL: http://atddoc.cern.ch/Atlas/DaqSoft/document/URD_27.html

[ATL00] The Atlas Collaboration, "High-Level Triggers, DAQ and DCS – Technical Proposal", *Technical Proposal*, CERN, Geneva, 2000, pp 165-168.

[BAR97] Antek Baranski, "Metadata", URL: http://wwwinfo.cern.ch/asd/cernlib/rd45/SlidesWorkshopJuly97/Metadata/index.htm

[GAP01] Igor Gaponenko et al., "The BABAR Database: Challenges, Trends and Projections", *Proceedings of CHEP 2001*, Beijing, China, 2001, pp 9-14.

[HOS01] Wolfgang Hoschek and Gavin McCance, "Grid Enabled Relational Database Middleware", presented at the *Global Grid Forum*, Frascati, Italy, 2001.

[JON97] Robert Jones at al., "The OKS in-memory persistent object manager", *IEEE Transactions on Nuclear Science, Vol.45, no.4, August 1998*, pp 1958-1964

[LUC02] Levi Lucio et al., "Test Report of the Online Book-keeper for the Atlas DAQ Online Software", *ATLAS DAQ Internal Note 177*, CERN, Geneva, 2002.

[MAL97] David Malon and Eduard May, "Critical Database Technologies for High Energy Physics", *Proceedings of the 23rd VLDB Conference*, Athens, Greece, 1997, pp 580-584.

[MOR97] G. Mornacchi et al, "The ATLAS DAQ and event filter prototype "-1" project", 10th IEEE Real Time Conference, Beaune, France, 1997

[NOW01] Marcin Nowak et al, "Object Persistency for HEP Data using an Object-Relational Database", *Proceedings of CHEP 2001*, Beijing, China, 2001, pp 272-275.

[OMG02] Object Management Group, "The OMG's CORBA website", URL: http://www.corba.org

[SHI98] Jamie Shiers, "Building a Multi-Petabyte Database: The RD45 Project at CERN", *Object Databases in Practice*, Prentice Hall, 1998, pp 164-176.

[SOL01] Igor Solovianov, "ATLAS Tile Calorimeter Run Information Database", URL: http://tileinfo.web.cern.ch/tileinfo/runinfo.php

[WEN02] Torre Wenaus, "Magda – Manager for Grid-based data", URL: http://atlassw1.phy.bnl.gov/magda/info

[1] A subdetector of the ATLAS, specialized in observing certain characteristics of the physics events.