HIERARCHICAL DATA ARCHIVAL SYSTEM FOR EMS

J.M. Nogiec†, K. Trombly-Freytag, D.G.C. Walbridge, FNAL, Batavia, IL 60510, USA

Abstract

The Extensible Measurement System (EMS) has been developed at Fermilab's Magnet Test Facility to allow for standardization of systems and to increase reuse of software. It is a component-based framework for building data acquisition and analysis applications. As a truly open system, EMS does not enforce one specific way of storing data and is capable of accommodating various archiving methods and formats. An application can be easily configured for many different archiving methods by using the appropriate components. The article presents one such solution, a two-stage, hierarchical storage system, which incorporates both object-oriented and relational databases. This solution accommodates the needs for performance, program-level access to data, and user-oriented access. The architecture and results of performance studies are presented.

1 INTRODUCTION

R&D environments impose specific and particular demands on test and analysis systems. These systems have to be flexible, extensible, and easily adaptable to ever changing requirements. Typically, the emphasis is put on the flexibility in data taking, user interfaces, and data processing, with the flexibility in data archival frequently overlooked. One of the goals of the EMS project was to build a system with storage architecture flexible enough to fulfill the following requirements:

- Several test systems in existence connected by a network or working temporarily as stand-alone systems
- Ability to store the system's state to continue processing later and to "replay" tests by streaming data from the storage rather than from the original data sources
- Capability to store results of ad-hoc configured systems and to store any intermediate results for reprocessing, debugging, alternative processing, and off-line processing
- Ability to sustain a relatively high data rate
- Independence of the archival mechanism from data processing, characterized by the capability to add new types of storage (files, databases, etc.) without modifying the existing solutions
- Ability to access and query final data from external applications

To accomplish these requirements the Extensible Measurement System framework [1][2][3] has been used together with a specially developed data archival solution.

†nogiec@fnal.gov

2 OVERVIEW OF EMS

Component-based systems seem to be particularly suitable for the R&D environment. They allow for relatively easy reconfiguration of the system and provide building blocks for constructing a whole family of systems. EMS exploits the component technology with its XML-based configuration language and bus-like communication architecture (see Fig. 1). Systems are built with EMS through explicit composition, by "wiring" components together to process system objects. Connections for control, exception, data, property, and debug objects are defined independently. The system allows also for the data-driven composition, where the execution of specific data processing algorithms is determined by the data themselves. The resulting flow of objects in the system can be viewed graphically.

The system is controllable via graphical user interfaces and scripting. Various components included with the framework (horizontal components) enable monitoring of the system resources, graphing, and run-time modifications to application components (so-called tailoring). Debugging and exception handling are built-into the system, with each component being able to produce exception and debug events. The level of detail that is output regarding the inner workings of the system can be set up in the configuration file and adjusted at the run time, which has proved to be extremely useful when constructing or troubleshooting systems.

The design of EMS does not restrict the system to archive only data. Through use of the software bus, information and control objects of any kind can be routed to appropriate components for viewing or storage. If one wants to keep a record of property, debug, exception, or even control objects, it is possible to direct them to databases and/or stores of other types.

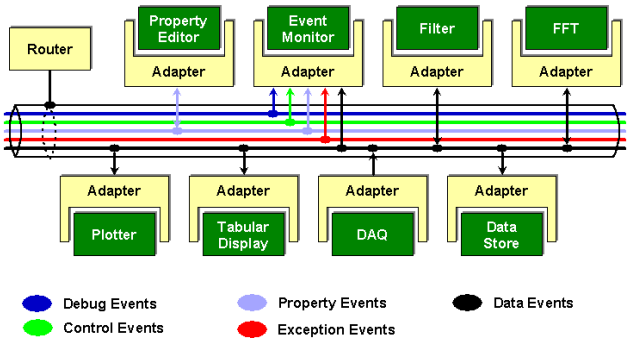

Figure 1: EMS architecture.

3 EMS ARCHIVAL SYSTEM

EMS, instead of choosing between a relational repository and an object repository, takes advantage of both of these representations by creating a hierarchical archival system (see Fig. 2). The lower level of the hierarchy comprises an object persistency mechanism whereas the upper level consists of a relational database.

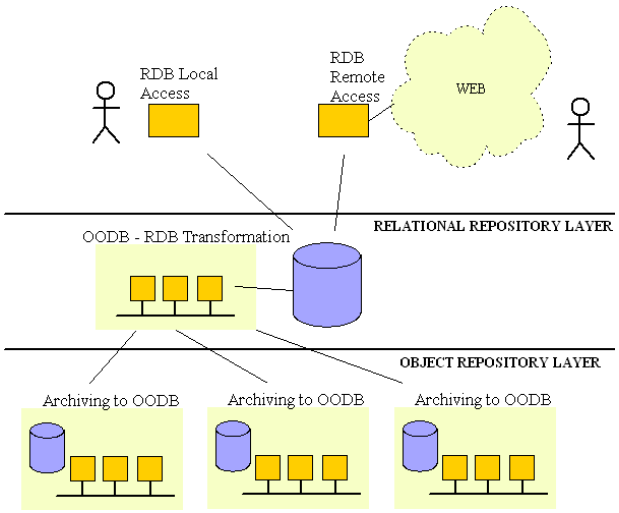

Figure 2: EMS systems and repositories

3.1 Object Repository

The object persistency mechanism, implemented as an object-oriented database, resides locally and provides mainly for programmatic access when reprocessing data, and storing the system's state. Since mapping between persistent data and objects is not required, this is ideal for programmatic access. Object persistency can focus on recording the history/trace of the system's run. Sequences of event objects created at various stages of the data processing chain (see Fig. 3) are recorded and can be later "replayed". This very general model allows for practically any object produced by the system to be archived. Since there are no a priori defined data models, decisions about what to archive can be made even at run time.

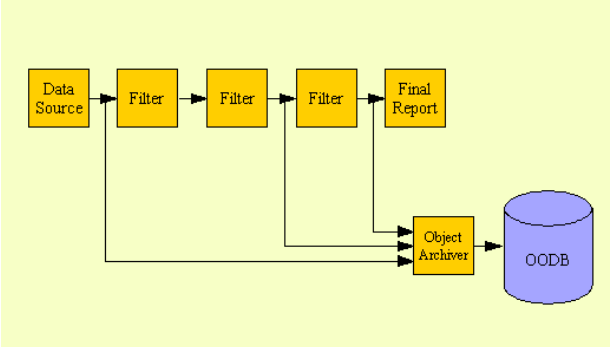

Figure 3: Archiving objects in the object repository.

The solution is general enough to accommodate various system configurations and objects originated by various components. In the model, sequences of related objects coming from a single system are called runs. Each run contains temporal collections of data, exception, and debug events. Persisted objects can be accessed selectively by defining the runs, originating component, and range of events. This allows for reprocessing to start from a known point in the data flow.

The object repository offers good performance with relatively high data saving rates. As shown in Figure 4, the saving rate depends highly on the size of objects persisted in a single transaction.

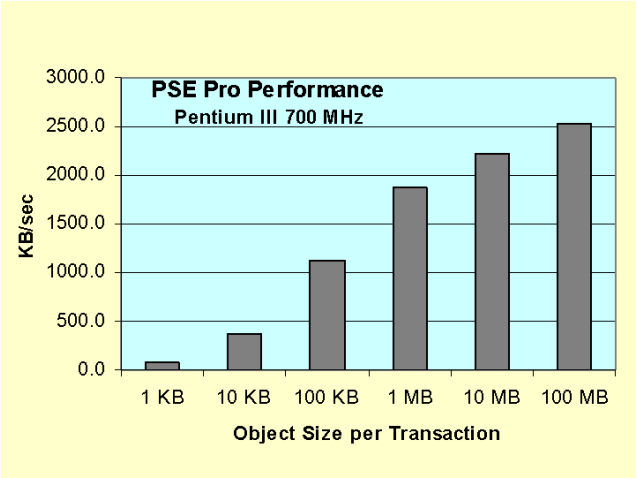

Figure 4: Performance of the object repository.

3.2 Relational Data Repository

Although perfect for a programmatic access, the object repository is less appealing for human interactions. Therefore, filtered, synthetic data that are ready for human interpretation are stored in the second layer of the EMS archival system, which is implemented as a relational database. The typical configuration, while having multiple object repositories, will have just one relational repository, which will be accessible remotely.

The data model of the relational database reflects concrete relations between data specific to a particular application and is foreseen, unlike the object repository, to change infrequently. Typically, this models the final results of the R&D process or production-type data.

Archival access to a relational database is provided by a flexible component using the JDBC standard. This component executes SQL queries and can provide data objects containing query results to the system. The query can be set up via the component's dedicated property or supplied in a data event. In the former case, the component can either load a stream of homogeneous data into a database using a predefined SQL insert statement or stream data out of the database using a given query statement. In the latter case, formal query parameters are substituted by string representations of the corresponding data items. Due to this flexible design, the database access component can perform SQL operations on behalf of another component as instructed in the received data event.

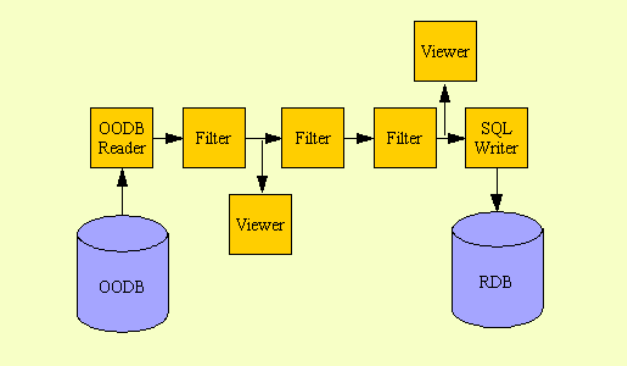

Figure 5: Archiving data in the relational repository.

3.3 Web Access

While remote relational database access works well for accessing data for on-line analysis, presentation of the analysis results for the human viewer is best done via the web. Templates for comprehensive reporting of test results as well as standardized result/test comparisons can be easily designed from a variety of existing solutions for displaying data stored in a relation database. Typical solutions include frameworks based on JSP and servlets or EJB. The EMS system will provide WEB access for the finalized data stored in the PostgreSQL relational database via JSP and servlets. A downloading capability for multiple formats of the data (e.g., the Excel file format) will be provided, as well as detailed reporting.

3.4 Implementation Variants

The current implementation of the EMS archival system uses an object-oriented database (the eXcelon's PSE Pro version of the Object Store database [4]) and a relational database system (PostgreSQL [5]). PSEPro is a pure Java database and therefore is as portable of the rest of the Java-based EMS. Additionally, it requires no administration assistance at all.

If the performance requirements allow for it, the native object-oriented repository could be replaced by the relational database with capabilities to store objects or an object persistency layer that hides the underlying storage mechanism and provides a mapping between objects and the actual storage system artifacts (such as tables). This mapping could be highly automated. Unfortunately, the solutions involving a relational database would require administration efforts and may also limit the portability of the system.

As far as the relational repository is concerned, various database management systems, including open source ones, could be used interchangeably as long as the necessary JDBC driver is available.

4 SUMMARY

A hierarchical archival system has been implemented to store data in R&D environments, where flexibility in defining stored objects as well as data selection and retrieval by humans are required. The system encompasses a primary storage implemented as an object persistency mechanism and a secondary storage implemented as a relational database.

This solution separates the user view from the programmatic view and allows for data filtering and reduction. Data taking is very fast due to performance of the object database and is not limited to performance of the relational database. No mapping of objects is required to achieve object persistency, which allows for keeping some ad-hoc generated results if needed for specific tests.

The presented solution avoids a typical dilemma associated with choosing between either exclusively relational or exclusively object model of data, with the relational approach being inflexible and the object approach being oriented more toward the programmatic access.

The described model of data archival has been implemented as part of the EMS project and successfully used at Fermilab's Magnet Test Facility.

6 REFERENCES

[1] http://sdsg.fnal.gov/emsweb
[2] J.M. Nogiec et al, "A Flexible and Configurable System to Test Accelerator Magnets", PAC'01, Chicago, 2001.
[3] J.M. Nogiec et al, "An XML Driven Framework for Test Control and Data Analysis", ICALEPCS 2001, San Jose, 2001.
[4] http://www.exln.com/products/psepro/
[5] http://www.ca.postgresql.org/