GENERIC REPOSITORY AND SEARCH ENGINE FOR LHC EQUIPMENT TEST DATA

M. Peryt, F. Momal
CERN, Geneva, Switzerland

Abstract

The construction of the CERN Large Hadron Collider involves an unprecedented amount of equipment testing. Very large volumes of data are taken by various data acquisition and SCADA systems and stored in plain files, in many formats and in diverse locations. Without appropriate tools, the domain experts have to put a lot of effort into locating the right information. Most of the time they have to transform data by hand in order to be able to analyze it with their favorite software tools.

We have developed a data storage framework addressing those issues. Data coming from various sources before their insertion into the repository are transformed to conform with an optimized, generic and open data model. System architecture is based on a three-tier paradigm to provide the separation between a storage layer and a data processing layer. Complex, user-defined queries and on-the-fly preprocessing of data are supported. Data access tools take advantage of new industry-standard technologies (WWW, Java). Thanks to this approach it is possible to access data in a platform-independent way and to plug-in the data access functionality into widely used software tools (spreadsheets, scientific toolkits).

1 INTRODUCTION

The environment in which the new archive facility operates is composed of many sources of information. We have to deal with data produced by various data acquisition systems, entered manually by operators or submitted by collaborating institutes and companies. There are a large number of distinct data formats. Files are stored in many locations, dispersed all over the CERN network. Consequently, it is not only hard to locate the right piece of information, but also to ensure the safety and good quality of data. On the other hand there is an ever-increasing demand for centralized storage of data and for consistent and easy to use search and retrieval facilities. Domain experts want to be able to retrieve and analyze the information in a user-friendly way, regardless of its origin. They do not want to be forced to perform several queries just because data in question was taken by different data acquisition systems. They wish to do statistics on data sets spanning months and years without having to browse tens of subdirectories on backup storage devices. They prefer to use industry-standard, versatile software tools to process and analyze data. They certainly would not mind should they be able to automate their routine, everyday tasks. Their task is to look *at* the information, not to look *for* it.

Our archive facility addresses those issues by providing a modular, multi-layer framework for archiving and for platform-independent retrieval of data in heterogeneous distributed computing environment. We chose not to deliver one more two-tier closed solution composed of a central database and surrounding user interfaces. We wanted a system that would be open enough to follow the inevitable evolution of information gathering systems related to the development of the LHC. We also wanted to be able to cope with the fast evolving new Internet technologies in order to take full advantage of facilities they provide.

Several groups at CERN already use the archive facility. Several new projects plan to include it in their data management infrastructure. The system undergoes constant development and new modules are being added on a continuous basis.

This paper discusses the requirements for the system, its architecture and plans for the future.

2 REQUIREMENTS

2.1 What we must store

The archiving facility must accept various formats of data acquired by test and measurement systems. In addition, it must be able to store many kinds of configuration information or calibration parameters. At lower level we can distinguish two basic flavors of data:

1. Time-stamped data. Those include stepwise-constant and fixed-frequency data. Sampling rates cover a wide range, from several minutes down to microseconds and even higher frequencies are expected in the future.
2. Tabular data. These are normally records composed of an arbitrary number of fields. The relationships and interconnections between records may be rather complex.

2.2 How we can search

The tabular data are searched with SQL-like queries. In most cases search criteria are combinations of logical, arithmetical and string operators. Time-stamped data are more difficult to search. We must provide access at various levels of granularity. It must be possible to retrieve a single value recorded at a certain point in time. On the other hand we must be able to work with very large sets of data, spanning weeks, months or even years (e.g. we want to know how many weeks a magnet spent below 1.9 K during the last two years). That means that we have to provide an efficient time-series management system.

2.3 *What we search with*

Gone are the days when one had to write a different user-interface for each supported platform. Fortunately, we do not have to dig anymore into details of numerous operating systems and GUI's. What we want to do is take advantage of software that is installed on almost every personal computer or workstation. And we want to let the people do their job even if they are away from their office. This is easily achievable through a Web service managing the data while integrating that service into popular data analysis software (spreadsheets, mathematical and statistical toolkits).

2.4 *What else we can do*

In addition to being a data store and search engine, our archiving system must also provide the necessary infrastructure for dynamic data processing. This includes on-the-fly transformations of raw data into physical values, statistical analysis, digital signal processing, etc. One way to achieve this is to define a specification for plug-ins that will be dynamically included into the archiving system engine to perform custom manipulations on raw data.

3 SYSTEM ARCHITECTURE

Figure 1 provides an overall view of the architecture of the archiving system. We can easily distinguish the three logical tiers, each of them containing several functional components. At the top there is a *client layer*, providing the user interface to request the services from the archiving system and to display the responses. The middle layer is composed of *application services* that interpret and dispatch user's requests and send back the replies. It is the layer that knows the logical structure and physical locations of data residing in the bottom layer of *data services*. Each flavour of data is assigned to a particular data store, optimised for efficiency when dealing with that particular kind of information.

When we look closely at the client layer, we can distinguish two groups of components. The first group supports a one-way data flow and deals with pre-processing and insertion of DAQ data into the archiving system. Typically it works in a batch mode with processes being launched periodically (eg. once every 24 hours) or upon request (after each burst of data). DAQ data files can be in any format. One way to interpret their content is to put them through some kind of input filter that converts them into intermediate format known to the archiving system. Then, they can be loaded with a generic loader module. An alternative to that solution is to group

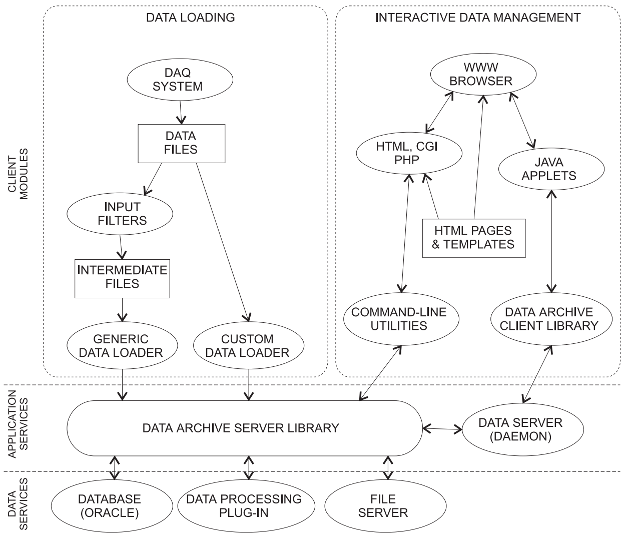

Figure 1. Overall system architecture

an input filter and a data loader into one module (custom data loader). The application server ensures that data entered into the system are coherent.

The components belonging to the second group within the client layer are interactive. They are controlled directly by a user. Their role is to provide the tools to search and retrieve the data from the archiving system and also to manually edit (insert, update, delete) the information. A significant effort has been put to implement those components using very standard tools (HTML, Java, server-side scripting, CGI), so that users do not need any exotic software to perform their tasks.

The application services layer is built around a common data archive server library. This library constitutes a unique interface with the repository. It ensures that all modifications to the archive move the data from one consistent state to another. The client layer modules can either link directly with the library to make use of the services it provides, or can communicate with the data server daemon using well defined protocol built on top of TCP/IP. The library itself interprets the client requests and redirects them after appropriate translation to the data server modules. This approach allows dissociating the data storage and data processing and thus provides optimised solutions to those issues.

Data services are as diverse as data flavours that the system has to deal with. Some data models yield themselves better to relational database storage, others fit particularly well into plain file systems, with indexes stored in a fully-fledged database. It is also necessary sometimes to generate information on the fly by applying certain procedures and parameters to raw data. All these services are provided by specialised modules belonging to the data services layer.

4 IMPLEMENTATION NOTES

The system has been developed using object-oriented techniques. OMT was used in the design phase. Most of the code, including the data archiving server library, was written totally in C++ and makes use of several commercial OO libraries. Even the interaction with relational database (Oracle™) is encapsulated into a library of objects. There are two versions of the data archiving client library: one written in C++ and the other in Java. The interoperability has been achieved thanks to the two-way mechanism allowing for serialisation and de-serialisation of Java and C++ objects. For the WWW gateway we use CGI scripts and PHP server-side scripting (http://www.php.net), as well as our own product: cs_gw (see [1]). We provide Java applets in the cases where more interactivity is required.

5 USE CASES

The following use cases describe two different occurrences in which our data archiving facility has been successfully used. The first system is the LHC test string, where we had to deal with time-stamped data and configuration information. The second one uses rotating coils to measure the magnetic flux as a function of coil angular position, producing thousands of records of data.

The purpose of the test string project is to measure the properties of assemblies of superconducting magnets. The cryogenics and vacuum installations are controlled by a commercial supervision system. A data acquisition system developed with LabVIEW™ is used to record the slowly changing data as well as the transient data from a number of transducers. In addition, geometrical data are produced on an irregular basis to record the information about minuscule movements of the assembly. Consequently, the test string data are coming from three sources. The first source is the supervision system, producing slowly changing logging data in form of comma-separated values (CSV) files. The second one is the data acquisition system, logging data and transient data in form of binary files in LabVIEW™ native format. The displacement sensors, whose CSV files are generated by specialized calculation routines, represent the last source. For each of those subsystems an input filter has been developed to convert data into a common intermediate file format. Those intermediate files are loaded into the archiving system. Users can retrieve the data through forms-based WWW interface. They can perform time searches or event (e.g. trigger) searches and freely combine channels belonging to any of the original sources of data. The resulting CSV files are transmitted to users' local computers for subsequent analysis. MS Excel™ integration using macro sheets is provided on MS Windows™ platform so that data are sent directly from Web browser to a spreadsheet and users are able to display charts or perform standard analysis procedures with a few clicks on their mouse.

The task of the archiving system, in the case of magnetic measurements, is to store the parameters of coils involved in the process as well as the raw data from measurements. Users can locate the relevant measurements by specifying a number of criteria and can then apply various analysis procedures in order to study the properties of magnetic field. Coil parameters are also involved in the analysis process as they allow the transformations of raw data into, for example, the information, which is necessary to perform the harmonics analysis. Users can manage the coils parameters with a Java applet. The retrieval of raw measurements is possible through forms-based WWW interface and a command-line utility running on most of the Unix platforms. The output of this utility can be easily piped into analysis tools provided by the domain experts to automate the standard analysis procedures.

6 CONCLUSIONS

The data archiving system has received a warm welcome among the users and is one of the basic tools in their everyday work. In addition, it is now becoming a part of a larger project whose objective is to provide an integration framework for various aspects of LHC controls. We aim at an automatic configuration of control, data acquisition, visualisation and archiving subsystems according to one generic description of the particular system. Presently each subsystem has to be developed separately thus possibly leading to some inconsistencies.

One of the subprojects of this integration framework is an advanced time series management library supporting very high data sampling frequencies and an extensive range of operations on time-stamped data. It is now in the design phase and the development should be completed in the first half of year 2000.

We are also working to replace the non-standard communication layer of the data archiving client and server libraries with code based making use of CORBA.

More and more industry-standard based interfaces into the archiving facility will be provided. We are currently looking into JDBC™ and ActiveX™.

REFERENCES

[1] F. Momal, , "A gateway between the Web and process control data (cs_gw)", http://wwwlhc.cern.ch/Docs/cs_gw_docs.htm

[2] M. Peryt, "Archiving System for LHC Test and Measurement Data. Technical Report.", http://wwwlhc/RPTS/ArchivingSystemReport.PDF