INTRODUCTION TO THE ANAPHE/LHC++ SOFTWARE SUITE

Andreas Pfeiffer
CERN, Geneva, Switzerland

Abstract

The Anaphe/LHC++ project is an ongoing effort to provide an Object-Oriented software environment for future HEP experiments. It is based on standards-conforming solutions, together with HEP-specific extensions and components. Data persistence is provided by the Objectivity/DB Object Database (ODBMS), while the visualisation is based on Qt (for 2-D presentation) and OpenInventor (for 3-D). To complement the standard package, a set of C++ class libraries for histogram management, Ntuple-like analysis (based on Objectivity/DB) and for presentation graphics (based on Open Inventor) have been developed. A new tool for physics data analysis, named Lizard, has been developed based on a set of abstract interfaces (as defined by the AIDA project). Its implementation is based on the python scripting language and the existing C++ class libraries of Anaphe/LHC++.

1. INTRODUCTION

The Anaphe/LHC++ project was set up to replace the full suite of functionality formerly provided by the FORTRAN based CERNLIB [1] in the new object-oriented computing environment required by (mainly) the LHC experiments.

The aim of the project is to provide a flexible, interoperable, customisable set of interfaces, libraries and tools, re-using existing (public domain or commercial) packages wherever applicable and possible. HEP specific adaptions are written where needed. In close collaboration with the experts from the experiments and other common HEP projects (like Geant-4 [2] and CLHEP [3]), these packages are designed considering the huge data volume expected for LHC, the distributed computing necessary to analyse the data as well as long-term evolution and maintenance.

The Anaphe/LHC++ [4] software suite consists of a set of individual class libraries for data storage (Objectivity/DB [5] and HepODBMS [6]), fundamental numerical mathematics (NAG-C [7] and CLHEP) and physics data analysis (HTL [8] for histogramming, Gemini/HepFitting [9] for minimization/fitting) which exist since about 1997 in their present form.

Recently, the development of packages for visualisation (Qplotter, based on the Qt [10] package) and a command-line driven interactive data analysis tool (Lizard) has started. The first release of these is scheduled for mid October 2000.

2. THE LIBRARIES

2.1 Data Storage

2.11 Packages

In order to study possible solutions for the storing and handling of the multi-Petabytes of data expected from the LHC experiments, a R&D project (RD45 [11]) was set up in 1995. The requirements included not only to store the raw data, but also the reconstructed objects, calibration and monitoring data and analysis objects like histograms and ntuples. After some study, it was found that the best candidate to fulfil the requirements is an Object Database Management Group (ODMG) [12] compliant database used together with a mass storage system (based on the IEEE reference model [13]). Several tests on available Object Databases were performed and resulted in the choice of Objectivity/DB for the time being.

On top of this, a small (about 15 kLines) s/w layer (HepODBMS) has been developed to simplify the possible porting to a different Database implementation and to provide a higher level interface than the one specified by ODMG. The HepODBMS class libraries provide classes to deal with session management, clustering hints, tag (ntuples) and event collections.

2.12 Special features

The use of an OO Database with transaction safety (locking) guarantees consistency in the datasets written. This is especially of importance in a distributed/concurrent environment.

Another important feature of the OO Database is location independency, meaning that moving the individual database files to other file systems and/or other hosts has no impact on the user's code.

2.13 Scaling to LHC

Meanwhile, significant experience in using Objectivity has been gained and scaling behaviour has been verified by HEP experiments like BaBar and CMS.

During the last year, the BaBar experiment has accumulated more than 100 TByte of data in Objectivity/DB and by this showing scaling behaviour to amounts of data which are only one order of magnitude lower than the ones expected for LHC. Another aspect of scaling is the number of processes used in processing (filtering and reconstruction), where BaBar is presently using 140 parallel processes for online reconstruction. This again is only about one order of magnitude lower than the requirements for LHC. Of course, one order of magnitude is still a “qualitative” step, nevertheless it shows that the underlying concept scales to quite a large fraction of the size needed for LHC.

In the CMS experiment several Data Challenges have been (and are) done where a few hundred processes were reading Geant simulation data, overlaying events and reconstructing the combined events in parallel. The simulated data is read from and the reconstructed objects are stored into Objectivity/DB. The overall sustained data rate achieved is about 70 MB/s. Analysis of this (reconstructed) data is done in parallel by a few tens of physicists using several hundred jobs on a distributed farm of PCs.

2.2 Fundamental physics and mathematics functionality

For fundamental physics and mathematics functionality, like dealing with 3- and 4-Vectors, simple and complex matrix operations and optimized numerical algorithms for special mathematical functions, the CLHEP and NAG libraries are used. CLHEP, a project started in 1992, aims at providing a fundamental set of C++ classes specialized for use in HEP, while the Numerical Algorithms Group (NAG) provides an optimized set of C functions dealing with a broad spectrum of mathematical (and physical) algorithms, e.g., for minimization, complex number functions, Fourier transforms, quadrature/integration, ordinary differential equations, curve and surface fitting, linear algebra, approximations of Special Functions (Bessel et al.) and more.

2.3 Histogramming

The Histogram Template Library (HTL) is a C++ class library that provides powerful histogramming functionality. As the name suggests, it exploits the template facility of C++ and is designed to be compact, extensible, modular and performant. As such it only deals with histograms - i.e. binned data - and not unbinned or ”Ntuple” data. Furthermore, although simple file-based I/O and ”lineprinter” output are supported via helper classes, it is decoupled from more advanced I/O and visualisation techniques. In the context of LHC++, such capabilities are provided by other components that fully interoperate with HTL.

HTL itself offers the histogramming features of HBOOK as well as a number of useful extensions, with an object-oriented (O-O) approach. This package replaces the histOOgrams package - an earlier C++ class library for histograms. The major functional innovation over the previous package are the support for different kinds of bins, the support of both persistent and transient (i.e. in-memory) histograms at runtime and the definition of an abstract histogram interface. As a result, it is possible to work with transient histograms and subsequently save some or all of them in a database as persistent histograms in a very simple and natural way (thus simulating so called explicit I/O). This clearly has significant performance advantages, particularly in the area of filling operations.

The definition of an abstract histogram interface allows functionality that is provided by external packages, such as plotting or fitting, to be decoupled from the actual implementation of the histogram. This feature paves the way for co-existence of different histogram packages that conform to the abstract interface.

2.4 Minimizing and Fitting

2.41 Minimizing

Minimization and Fitting in the context of Anaphe/LHC++ is presently in a major re-design phase. For a detailed description of the new design, see [14]. As the new packages make extensive re-use of the previous ones, the latter are described in some detail here.

Gemini is a GEneral MINImization and error analysis package implemented as a C++ class library. Minuit's functionality is provided in a 'Minuit-style' (even if, internally, another minimizer may actually do the work) and new functionality offered by NAG C minimizers is added. Gemini thus provides a unified C++ API both to standard Minuit and to NAG C family of minimizers. For the common subset of functionality, it is up to the user which minimization engine does the work: Minuit or NAG C. The user can easily switch between various minimizers without essential changes in the application code. The currently supported set of minimizers (Minuit and NAG C) can be extended without essential changes in the API.

2.42 Fitting

HEPFitting is a collection of C++ fitting classes, based on the general minimization package Gemini. It provides an object oriented application programming interface, which allows for loading data, defining a model and a fitting criterion, performing a fit in a specified region and obtaining fit results, including error analysis.

Histogram data to be fitted to can be loaded using the HTL package. Alternatively, general data points of the form (x, y) and the corresponding errors can be loaded via user's arrays, where y stands for an experimental value measured at a point $x = (x1, ..., xp)$, and a user model $y = f(x1, ..., xp|\mathbf{a})$ can be fitted to the loaded data, resulting in an estimate of the parameter vector $\mathbf{a}$.

2.5 Visualisation

2.51 2-D Visualisation

In spring 2000, the decision was made to change the visualisation of 2-D data to use the Qt package instead of the previously used OpenInventor package. Qt is a multi-platform C++ GUI toolkit, produced by Troll Tech, it is supported on all major variants of Microsoft Windows and Unix/X Windows. Qt is the basis of the popular KDE desktop environment on Linux.

On top of this package, a small set of additional functionality (specific for HEP) has been added in the form of the Qplotter package. The Qplotter library allows a programmer to produce graphic representation of physics data (such as histograms, scatter plots or curves) both on the screen and as PostScript files. The same package could be used to produce simple 2D drawings (e.g. test beam setup), but it will not provide directly event display features, since they can be implemented using 3D packages like OpenInventor/OpenGL. Nevertheless the library should eventually allow mixing both kind of images in the same viewer.

Full user interactivity (such as point-and-click or drag-and-drop) will not be pursued from the beginning, although it may be eventually supported by add-on tools.

2.52 3-D Visualisation

The low-level 3D graphics is based on the OpenGL library, the de-facto industry standard. The OpenGL rendering process is highly optimised in order to obtain satisfactory performance. In general, OpenGL is implemented in low-cost hardware accelerating graphics cards. The high-level 3D graphics is based on OpenInventor [15], which is a high-level C++ class library based on OpenGL.

The foundation concept in Inventor is the ”scene database” which defines the objects to be used in an application. When using Inventor, a programmer creates, edits, and composes these objects into hierarchical 3d scene graphs (i.e., database). A variety of fundamental application tasks such as rendering, picking, event handling, and file reading/writing are built-in operations of all objects in the database and thus are simple to invoke.

Since Inventor is object-oriented (written in C++), it encourages programmers to extend the system by writing new objects. Inventor users have created a variety of new objects that are not included in the product, such as: Bezier surfaces, animation objects, special viewers, and many more.

Although presently none of the other packages provided by Anaphe/LHC++ is making use of 3-D graphics (OpenInventor), we provide and maintain OpenInventor for users who want to use it.

3. ABSTRACT INTERFACES FOR DATA ANALYSIS – THE AIDA PROJECT

The goals of the AIDA project [16] are to define abstract interfaces for common physics analysis tools, such as histograms. The adoption of these interfaces make it easier for users to select and use different tools without having to learn new interfaces or the need to change their code. In addition it should be possible to exchange data between AIDA compliant applications. The developers of the following packages and analysis systems are actively contributing to, interested in, or plan to adopt AIDA: COLT, JAS, Lizard, Open Scientist.

A set of categories has been identified for which *independent abstract interfaces will be defined. These components contain the basic data structures used in physics data analysis like Histograms, Ntuples, Functions, Vectors as well as higher level components like Fitter, Plotter, Analyzer, Event-Display and Controller/Hub.*

At the time of writing this report, this concept of abstract interfaces in combination with a number of different actual implementations has been proven and verified. In collaboration with GEANT-4 examples of simulations have been set up using the AIDA Histogram interfaces creating and filling histograms which were implemented using three different analysis systems (JAS, Lizard, Open Scientist) without the user-code being aware of it.

4. INTERACTIVE DATA ANALYSIS TOOL – THE LIZARD PROJECT

Lizard [17] is a new Interactive Analysis Environment created within the Anaphe/LHC++ context at CERN. The aim of the Lizard project is to produce an analysis tool which can be easily integrated in a C++ based environment and which provides functionalities at least comparable with the core of PAW.

4.1 Architecture and Design

Lizard is aiming at being a highly modular, flexible, interoperable and customizable tool. It is based on the set of components identified for AIDA (see above). Until all the interfaces have been agreed within *the AIDA group, a separate set of interfaces has been defined. This set will be subsequently replaced by the ones defined in AIDA as they become available.*

The modularity and flexibility is achieved by a loose coupling between the individual components through their abstract interfaces. By doing this, each component is only allowed to access another through its (abstract) interface, therefore no knowledge about the implementation is needed at compile or link time (the latter due to the use of shared libraries).

This approach also ensures a high degree of customizability, as an experiment can implement their own version of any component (defined through the abstract interface). In the same way, components from other analysis systems can be used by simply loading the corresponding set of shared libraries from the other system (and possibly some "bridging code", as in the case of JAS).

4.2 Scripting

On request of the users, Lizard is using a command line driven approach. To implement this, the choice was made to use one of the various existing scripting languages as the main use cases showed to be more similar to the typical ones for scripting (repetition, macros) rather than (algorithmic) programming.

In order to keep maximal flexibility, it was decided to use SWIG [18] to "map" the commands (which are defined as a set of C++ classes with their methods) into one of several possible target scripting languages. The choice of the scripting language was Python [19] as it seemed to be the most likely one to be accepted and the fact that it is object oriented by design.

4.3 Status of Lizard

A first prototype with rather limited functionality of Lizard has been released in Feb. 2000. This prototype was basically intended to check the possibilities of the automatic "mapping" of commands into python, as well as gaining user feed-back regarding the choice of Python.

In October 2000 a first version of Lizard has been released as scheduled. Work is continuing to improve the functionality and add more features. Further releases are planned in intervals of about 4 weeks.

The study of the impact of a distributed computing environment on data analysis - as it is expected for LHC - will start in 2001. The results of this study will then be integrated into Lizard. The aim here is to create a tool which can do parallel data analysis using a large "farm" of machines (on typically a few TeraByte of data), or "small scale" analysis on a single machine (typically several GigaByte of data) with basically the same "front-end" user interface.

The "farming" environment is expected to be based on the GRID infrastructure which is presently under study and development at CERN.

5. FUTURE PLANS

5.1 Libraries

The main work on the libraries will be the implementation of wrappers to the abstract interfaces as used in AIDA and Lizard. This is done using existing class libraries wherever possible.

5.2 Lizard

Future work on Lizard can be split into the following two groups:

Near future:

- work on documentation
- direct use of Python objects (functions, list, ...)
- more plotting features

- “license free” version of Lizard
- reading of HBOOK files (histograms and ntuples)
- history recall across sessions
- XML format for histograms and vectors
- finalise Fitter interface

Longer term:

- analyse/design/implementation of distributed computing using Globus (GRID)
- allow communication between components using an Observer pattern (“live histograms”)
- more plug-in-like style to allow for several different instances of the various components to be present at a given time

5.3 AIDA

The main activity within the AIDA project will be to define and consolidate the interfaces to the other components. In the longer term, more functionality will be added, for example to allow for “live” histograms which are useful in online and monitoring environments.

6. Conclusion

The Anaphe/LHC++ software suite provides a flexible, interoperable, customisable set of interfaces, libraries and tools for physics data analysis. It is based on standards-conforming solutions, together with HEP-specific extensions and components. Data persistence is provided by the Objectivity/DB Object Database (ODBMS), while the visualisation is based on Qt (for 2-D presentation) and OpenInventor (for 3-D).

In close collaboration with the experts from the experiments and other common HEP projects, these packages are designed considering the huge data volume expected for LHC, the distributed computing necessary to analyse the data as well as long-term evolution and maintenance.

A new tool for physics data analysis, has been developed based on a set of abstract interfaces (as defined by the AIDA project). Its implementation is based on the Python scripting language and the existing C++ class libraries of Anaphe/LHC++.

In the future, we will - besides further evolution of the existing packages - start to investigate the impact of distributed computing on physics data analysis with the aim to make this type of analysis as transparent as possible for the user of the libraries and tools.

References

[1] CERN Program Library (CERNLIB); see http://wwwinfo.cern.ch/asd/index.html

[2] Geant-4 - A toolkit for the simulation of the passage of particles through matter. see http://wwwinfo.cern.ch/asd/geant4/geant4.html

[3] CLHEP - Class Libraries for HEP; see http://wwwinfo.cern.ch/asd/lhc++/clhep/index.html

[4] LHC++; see http://wwwinfo.cern.ch/asd/lhc++/index.html

[5] Objectivity/DB; see http://www.objectivity.com

[6] HepODBMS; see http://www???????

[7] Numeric Algorithms Group; see http://www.nag.co.uk

[8] HTL; see http://wwwinfo.cern.ch/asd/lhc++/htlguide/htl.html

[9] Gemini/HepFitting; see http://wwwinfo.cern.ch/asd/lhc++/Gemini http://wwwinfo.cern.ch/asd/lhc++/HepFitting

[10] Qt (Qt is a trademark of Troll Tech); see http://www.troll.no

[11] RD45 - a persistent object manager for HEP; see http://wwwinfo.cern.ch/asd/rd45/index.html

[12] The Object Data Management Group (ODMG); see http://www.odmg.org

[13] The IEEE Storage Systems Standards Working Group; see http://www.ssswg.org

[14] J.T.Moscicki, “Minimization and Fitting in Anaphe”, these proceedings.

[15] Open Inventor 3d Toolkit; see http://www.tgs.com/Products/openinv-index.html

[16] AIDA - Abstract Interfaces for Data Analysis; see http://wwwinfo.cern.ch/asd/lhc++/AIDA/index.html http://aida.freehep.org

[17] Lizard; see http://wwwinfo.cern.ch/asd/lhc++/Lizard/index.html

[18] SWIG - Simplified Wrapper and Interface Generator; see http://www.swig.org

[19] The Python scripting language; see http://www.python.org