Root For Run II

P. Canal[1], *S. Panacek*[1], *P. Malzacher*[1,2]

[1] Fermi National Lab, USA
[2] GSI, Germany

Abstract

Following a comprehensive understanding of the requirements for Physics Analysis, Fermilab adopted the use of ROOT for the Collider Experiments, CDF and D0 in the upcoming data-taking run - Run II. To meet the needs of the experiments, Fermilab has collaborated with the Root Development Team. After holding a workshop for ROOT users, the Fermilab Computing Division agreed to contribute to the development and support of ROOT. In this paper, we will review the contributions we have made and plan to make to ROOT. These include: The script compiler, the integration of CLHEP's physics vector, and the self-describing file format.

An important requirement for D0 and CDF is to have a robust and maintainable software tool for analyzing physics data. We have worked closely with the ROOT development team, testing and building ROOT for robustness and maintainability. We have also developed two hands-on courses and a user's guide introducing ROOT to the novice user. These courses and document go over both, the initial steps of using ROOT as well as features that are more advanced.

Keywords: ROOT,Data Analysis,RunII,I/O

1 Requirements

Before adopting software for Physics Analysis for the upcoming Run II, Fermilab and the Collider Experiments, CDF and D0 gathered a set of requirements. They covered two main areas, the functional requirements, and the support and maintenance requirements. The software tool needed to be able to access, analyze, and present (in reports and publications), the exceptionally large volume of data produced by Run II. Some other requirements are:

- A scripting language giving access to objects (as opposed to some simpler structure, such as arrays of numbers).
- The ability to run on several platforms
- A modular architecture, so that parts can be replaced and more parts can be added

The other requirements were captured in a questionnaire attempting to quantify product support, portability and ease of maintenance. There were also questions on how feasible it would be to collaborate with the product developers. This point was particularly important for freeware packages and software developed in the HEP community. In particular, it was noted that in making local additions or modifications to a product, one runs the risk of diverging from the code supported by the product supplier. A close two-sided collaboration was an absolute requirement.

The candidates for such a tool were Histoscope, LHC++, IDL, and ROOT. We found that ROOT was the closest to meeting our requirements. Histoscope lacked the scripting language, it would have taken a year or more to implement one. LHC++ was scheduled to be used in 2005 and was incomplete, and IDL had limitations on the size of data sets. In the end, we chose ROOT as the best tool available to do the job, however, some requirements were only partially met.

2 Workshop

The ROOT team enthusiastically welcomed our request for collaboration, and it was kick-started by a short but very fruitful visit of the new Fermilab ROOT support team to CERN, during which a few outstading issues were already resolved. Since several other US HENP Experiments were considering ROOT, Fermilab decided to hold a US HENP ROOT Users Workshop in March '99. During this sucessfull workshop developers and users agreed on a list of the most urgent features:

- Improved robustness and stability, especially in the I/O subsystem
- Automatic handling of STL containers and strings
- Improved documentation
- Interface to the CLHEP library
- Multi-threading
- Interface to SQL

3 Workshop Follow-up

3.1 Improved robustness and stability, especially in the I/O subsystem:

The External Preprocessor For CINT

One of the complaints about CINT was the limitation of its C preprocessor. It can expand C macros only on two levels. To circumvent these limitations, CINT is capable of using an external preprocessor. However, using an external preprocessor did not work when CINT was used with ROOT. The Fermilab ROOT team updated both ROOT and CINT to enable this feature.

The Modularization of ROOT

The CERN ROOT team has modularized the source code into shared libraries starting with the 2.23 release. Now, the libraries are loaded when they are needed, and when users are designing ROOT applications they can limit the number of ROOT libraries linked in.

The Script Compiler

One of our major contributions is ROOT's Script Compiler. The Script Compiler gives the ROOT user the choice of compiling a ROOT macro rather than using CINT and interpreting it. It compiles the ROOT macro and produces a shared library. The advantages are:

- The execution is about five times faster because the code is compiled rather than interpreted.
- It provides C++ capabilities beyond what CINT provides. For example it supports templates and the STL library to the full capability of the compiler
- Syntax checking is more thorough than CINT, and the error messages are more descriptive.
- It encapsulates all the steps needed to make a C++ source file available to ROOT. It can be used as a substitute for relatively complex makefile rules.

On the other hand, it takes longer to load the macro with the Script Compiler because of the compilation time. In addition, a macro running with the Script Compiler can not be reloaded because a C++ shared library can not be unloaded and reloaded. The compiler and compiler switches used by default are the same as those used to build the current root executable (the one from which the script compiler is invoked). These default values can be easily changed.

User Support, Deployment and Testing

We were concerned about being stuck with features that were implemented in a way that did not match the needs of the experiments, or with ‘features’ which we considered to be bugs and which could handicap us. To address this, we take two initiatives. First, we educate users on techniques and features they may have not been aware of. Second, we act as the central point of contact for major problems encountered by the experimenters. We provide help in determining if a particular behavior is due to a misunderstanding or to a bug in ROOT. We also gather feature requests and advocate them to the ROOT developers. Being the point of contact has worked out well. We have been able to study ROOT in detail, probably more so than the casual ROOT user. This allows us to quickly understand the problem and often provide a patch for it. In addition, since we represent a larger number of experimenters, we have a strong voice in discussing improvements and new features. We found the CERN ROOT team very eager to collaborate, and were pleasantly surprised at their willingness to take our contributions into consideration.

We wanted to insure a high consistency of the ROOT releases used by the experimenters of Run II. To do so, we use Fermilab’s product deployment tool (UPS/UPD), and we run and analyze the ROOT test on all the Run II platforms.

Automatic Handling of STL Containers and Strings, I/O improvements

The ROOT team has improved the ROOT file format to include a byte count that enables graceful recovery from reading unrecognized objects.

The Script Compiler added the capability of using STL containers. In addition, rootcint, the ROOT dictionary generator, was upgraded by the CERN ROOT team to properly handle STL containers and strings.

3.2 Improved Documentation And Education

Elaine Lyons, a summer student and wife of a visiting physicist, was tasked with writing a “Getting Started with Root” document. She wrote an excellent document by interviewing ROOT and PAW users, and by learning ROOT herself. She used a novel approach to getting users started by simply telling them how to use the GUI, ignoring the command line for the first chapter. In no time, the new user was opening files and looking at histograms. From the ”Getting Started” document, a hands-on Root Class (ROOT 101) evolved. The class has several short lectures followed by exercises to be completed in class. We had two instructors, one to speak and the other to demonstrate on the projected computer screen the task. It was very successful; we taught 140 people in classes of 12 - 20. We also developed ROOT 102. We first taught it in November ’99 and up to now have taught about 60 people. Its format is the same as ROOT 101 with emphasis on building and reading Root trees. When developing ROOT 102 we found we needed a basic introduction to C++ for ROOT users. We put one together and made it available on the website but did not teach it.

The statistics show that the C++ introduction received almost as many visitors as the ROOT 102 class. The statistics for the website are: 2537 hits for Root 101, 645 hits for Root 102, 532 hits for the C++ basics. The classes saved us time answering the same basic questions, it saved the new ROOT user frustration and the time to find basic but buried information, and it also helped ROOT’s popularity. The tutorial’s website is at: `http://www-pat.fnal.gov/root`

3.3 Interface to CLHEP

The Fermilab ROOT team provided ROOT with a new copy of the Vector package from CLHEP. A few additional methods were needed for backward compatibility. We will continue to implement developments in the CLHEP PhysicsVector classes in ROOT. This duplication of code is necessary because Physics Vectors are a very frequently used HEP concept; therefore, ROOT needed to provide an implementation without requiring the user to download CLHEP. So rather than importing the complete CLHEP source code, only the Vector package was imported. Although the interface is similar, we adjusted the class names for ROOT to avoid any confusion. We also plan to introduce a package that will depend on both ROOT and CLHEP. It will provide a set of routines to access all the CLHEP classes from ROOT command line, and save them in ROOT files.

3.4 Multi-threading, Interface to SQL

Thanks to contributions from GSI, multi-threading will be available soon. The CERN ROOT team added an interface to SQL.

3.5 Other Fermilab ROOT Development Projects

In addition to the requests from the workshop, we wanted to contribute other development ROOT.

Automatic Documentation

The ROOT's automatic documentation feature (the THtml class) uses the header and implementation files of a class and produces HTML describing the interface and offering links to source files. We saw that inline functions cannot be documented and local variables are not recognized and thus they do not have a link to their class definition. In addition, the information provided could be improved. We plan to add a click-able inheritance diagram, and a more flexible organization of the list of member functions. For example, we plan to add a view where all member functions, both inherited and local, are listed together.

Self-describing File Format

ROOT users can save their own objects to ROOT files. The objects are written to and read from file by the Streamer method. This makes it necessary to have access to either the source code or a shared library supplying the Streamer. If the Streamer is not available, for example if the source code was lost or changed, the objects can not be read from the file. We are working with the ROOT developers to change the ROOT file format to be self-describing. This means that along with the object, a description of its Streamer is saved to the file. It will enable the reading of the object data members without having access to the object's source code. Note that saving an object in a split-mode ROOT Tree already provides this functionnality.

3.6 Conclusion

A year after making the decision to use ROOT for HEP analysis in Run II, we have more confidence in our decision and have addressed the issues presenting the greatest risk. What made it work? We think these were the main ingredients:

- The unprecedented support by the CERN ROOT team. In our experience commercial vendors cannot provide such customization and support, not to mention direct access to the developers/architects.
- The makeup of the skills and personalities of the Fermilab ROOT team. It was crucial to have technical talent that could contribute code so that the CERN ROOT team did not only receive requests but also solutions. It was also important to have communication skills to educate and help users.
- The specific issues generated at the workshop, and the timely follow-up. Having specific issues gave us something to work on, and improving the robustness and stability of the ROOT reduced not only our risk but also made ROOT a better candidate for other experiments.