PROCEEDINGS OF SCIENCE

FairRoot Framework

Mohammad Al-Turany*
GSI DARMSTADT
E-mail: m.al-turany@gsi.de

Florian Uhlig
GSI DARMSTADT
E-mail: f.uhlig@gsi.de

FairRoot is the simulation and analysis framework used by CBM and PANDA experiments at FAIR/GSI. In this paper the newest developments in the FairRoot framework will be discussed. In detail these are the CMake based building and testing system, a new event display based on EVE-package from ROOT and Geane and finally the new developments for using GPUs in the event reconstruction will be introduced

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7, 2008
Erice, Italy

*Speaker.

PoS(ACAT08)048

1. Introduction

The FairRoot framework [1, 2, 3], is an object-oriented simulation, reconstruction and data analysis framework based on ROOT [4], and the Virtual Monte-Carlo (VMC) interface [5]. It includes core services for detector simulations and offline analysis. The framework, is designed to optimize the accessibility for beginning users and developers, to be flexible (i.e. able cope with future developments), and to enhance synergy between the different physics experiments at/or outside the FAIR project. FairRoot supports many systems and compilers. The nightly build system compiles and test the status of the project on many different platforms. The accumulated results from each of these platforms are send to a web server. The results will be displayed on dashboards where all developers can immediately see if there is any problem on any of the different systems. These tasks are achieved by using the open source tools CMake, CTest and CDash [8] with a set of macros that define certain tests. In Figure 1 the different implementations of FairRoot at different experiments are shown [1, 3, 6, 7].

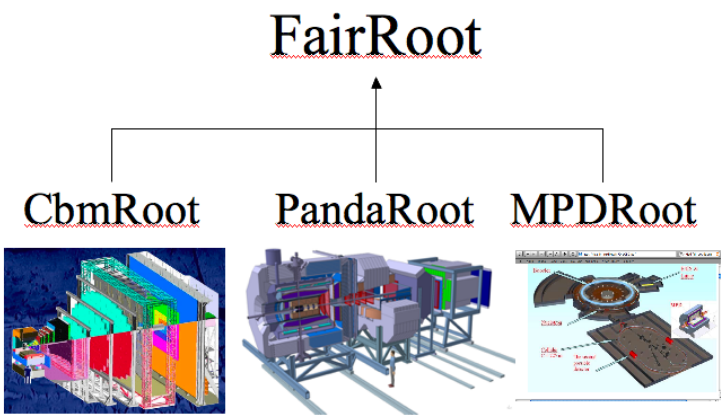

Figure 1: FairRoot at different experiments

2. Building system and quality control

One of the challenges of software development for large experiments is to manage the contributions from developers distributed among the different institutes and countries. In order to keep these developers synchronized a quality control is important. For a software project this means that it has to be tested on all supported platforms, e.g: If the project can be build from the sources, if it runs and in the end if the program delivers the correct results. This tests should be done frequently which results immediately in the necessity to do these checks automatically. If the number of different platform increases it becomes impractical to have installations of all supported platforms at one site. To overcome this problem, the best way is to use a client server architecture, which means to run the quality control at the place where a specific platform is installed and used (client) and only the results are send to a central server responsible for the processing of the data.

2.1 Build Environment

Another important point is the question how to create the Makefiles for the different platforms. With increasing number of platforms it becomes unmanageable to write the Makefiles by hand so one needs an automatic way to produce the Makefiles. CMake is an extensible, open-source system that manages the build process in an operating system and compiler independent manner. Simple configuration files are used to generate standard build files (e.g., Makefiles and/or projects/workspaces: Xcode, Eclipse CDT4, Kdevelop3, etc) which are used in the usual way depending on the platform and development environment. As part of CMake comes the tool CTest which is used to do all the checks. As first step it automatically updates the local working copy from the SVN repository before it runs the configuration which creates the Makefiles and then builds the project. As last step CTest runs the defined tests and send all the generated log data to the central web server. To test the project the same ROOT [4] macros are used as for the normal simulation and reconstruction. The test stage is easily extendable by adding new macros. With CMake/CTest it is also easy to execute other external programs. For example the class documentation is generated automatically on a daily basis using Doxygen [10].

2.2 Rule Checker

FairRoot makes use of a RuleChecker [9], which is a tool for checking the compliance of C++ code with the experiment coding conventions. It is a product developed in collaboration between IRST institute and with the ALICE and ATLAS collaborations. The tool can be customized to check a certain number of experiment coding conventions. This code check is included in the CMake environment of FairRoot which makes it easy to run the checks and to create in the end a nice web page which can be used by the developers to find violations of the coding conventions[11].

2.3 Dashboard

The second part of the quality control is to collect all the produced information, to process them and as final result to display them in an easy to use fashion. For these tasks the software package CDash is used which creates after processing the incoming data as a result linked web pages which display all the gathered information in a so called Dashboard. From here all developers can easily access the generated information.

3. Event Display

The event display in FairRoot is based on the EVE (Event Visualization Environment) package in ROOT [4]. Combined with trajectory visualization in FairRoot, the event display can be used directly from macro to display TGeoTracks (MC Tracks), Monte Carlo points and hits, together with the detector geometry. The FairEventManager implemented in FairRoot deliver an easy way to navigate through the event tree and to make cuts on e.g. energy, pt or particle ID in user events. However the drawback of this method is that the tracks has to contain some visualization information in order to be displayed. In the following we describe a solution where the track information is created at visualization level without the need to store track information during the simulation. This is achieved using GEANE.

3.1 GEANE

GEANE [12, 13] is a package to calculate the average trajectories of particles through dense materials and to calculate the transport matrix as well as the propagated errors covariance matrix in a given track representation. GEANE has been fully integrated in FairRoot as a package. The framework defines the basic classes relevant for track following, i.e. configuration, geometry description (from the Monte Carlo, parameters files) and the magnetic field map definition. The exact geometry and field used in the simulation can be taken into account by the track follower.

3.2 Event Display with GEANE as track propagator

The fact that both EVE and GEANE uses the ROOT geometry description for detector geometry make it natural to integrate GEANE as a track propagator for the event display. With this integration reconstructed tracks and/or Monte Carlo tracks can be visualized to a good accuracy. The main advantage of this integration is that any set of reconstructed or simulated tracks can be visualized without the need of special visualization modes. Moreover only selected tracks will be propagated on the fly which enhance the performance of the display. Figure 2 shows a schematic diagram of the interface to GEANE.

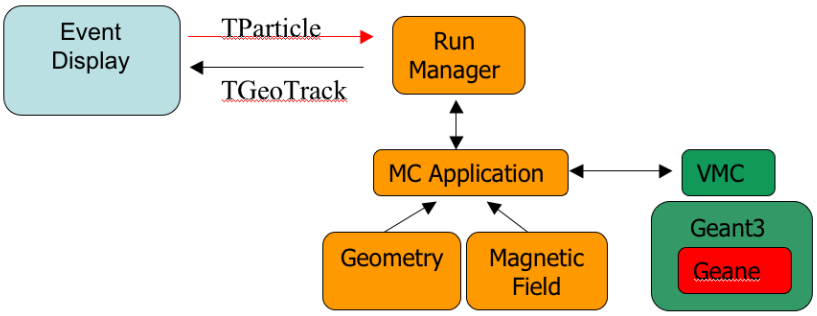

Figure 2: Event Display and GEANE

4. GPU's and CUDA

In the last few years, the graphics processor units (GPUs) have moved away from the traditional fixed-function 3D graphics pipeline toward a flexible general-purpose computational engine. Moreover they are getting cheaper and more powerful [14]. With the Nvidia Compute Unified Device Architecture (CUDA) [15], one can get orders-of-magnitude performance increases over standard multi-core processors, while programming with a high-level language such as C[14].

CUDA, is freely available and the CUDA development tools work alongside the conventional C/C++ compiler, so one can mix GPU code with general-purpose code for the host CPU. CUDA automatically manages threads, i.e. Does not require explicit management for threads in the conventional sense, which greatly simplifies the programming model. However, developers must analyze data structure and determine how to divide the data into smaller chunks for distribution among the thread processors.

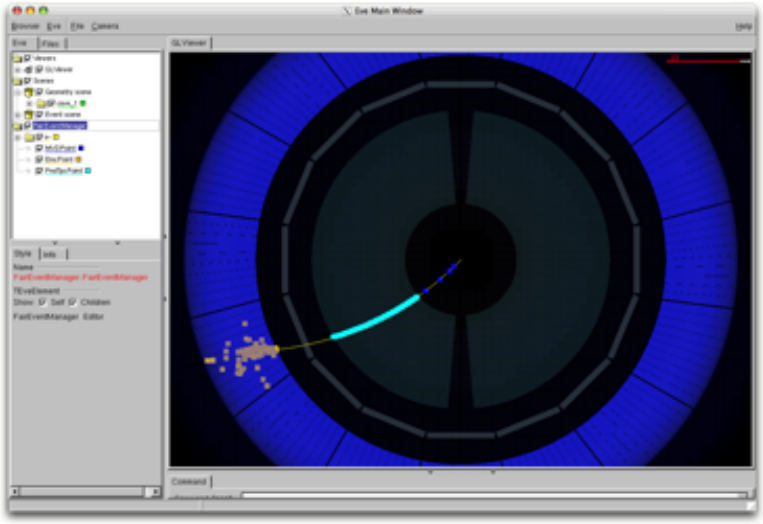

Figure 3: Event display example: Panda inner tracker and EMC

4.1 CUDA integration into FairRoot

Using FindCuda.cmake [16] CUDA is integrated into FairRoot building system very smoothly. The users do not have to take care of Makefiles or which compiler should be called (e.g. NVCC or GCC). Moreover an interface is under development which enables the user to use the GPU from within a ROOT CINT session, this interface will be published and available soon this year.

References

[1] M. Al-Turany, D. Bertini, and I. Koenig. CbmRoot: Simulation and analysis framework for CBM experiment. In S. Banerjee, editor, *Computing in High Energy and Nuclear Physics (CHEP-2006)*, volume 1 of *MACMILLAN Advanced Research Series*, pages 170–171. MACMILLAN India, 2006.

[2] D. Bertini, M. A-Turany, I. Koenig, and F. Uhlig. The fair simulation and analysis framework. In *International Conference on Computing in High Energy and Nuclear Physics (CHEP'07)*, volume 119 of *Conference Series*. IOP Publishing, 2008.

[3] M. Al.-Turany FairRoot: http://fairroot.gsi.de.

[4] R. Brun and F. Rademakers. Root - an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research A*, 389:81–86, Sep. 1997.

[5] R. Brun, F. Carminati, I. Hrivnacova, and A. Morsch. Virtual Monte-Carlo. In *Computing in High Energy and Nuclear Physics*, pages 24–28, La Jolla, California, 2003.

[6] PANDA Computing Group in GSI Sci. Rep. 2006, FAIR-EXPERIMENTS-02

[7] http://nica.jinr.ru/

[8] I. Kitware. Cmake: www.cmake.org.

[9] A. Potrich and P. Tonella. Itc-irst c++ code analysis: an open architecture for the verification of coding rules. In *CHEP 2000*, pages 758–761, 2000.

[10] Doxygen http://www.doxygen.org.

[11] M. Al-Turany, D. Bertini, and F. Uhlig. Improving the software development environment in FairRoot. Instruments-methods 60, GSI scientific report, 2007.

[12] V. Innocente, M. Maire, and E. Nagy. *GEANE: Average Tracking and Error Propagation Package*. CERN, Geneva, 1994.

[13] A. Fontana, P. Genova, L. Lavezzi, A. Panzarasa, A. Rotondi, M. Al-Turany, and D. Bertini. Use of geane for tracking in virtual monte carlo. *Journal of Physics*, 119(032018), 2008.

[14] CUDA, Supercomputing for the Masses, http://www.ddj.com/hpc-high-performance-computing/

[15] NVIDIA http://developer.nvidia.com/object/cuda.html

[16] Abe Stephens http://www.sci.utah.edu/ abe/FindCuda.html