R3BRoot / FairRoot

Dmytro Kresan (GSI-IT)
How it started?

- We need simulations for the LOI
- We have no manpower for software
- Re-use existing software
- It has to be easy, fast, reliable, ..etc
- We need it yesterday
How a framework can help?

- Allows physicists to concentrate on detector performance details, avoiding purely software engineering issues like storage, retrieval, code organization etc;
- Do not submerge into low-level details.
- Use pre-built and well-tested code for common tasks.
FairRoot was officially created in 2006

- CBM and PANDA invest manpower in the core team
- The GSI decided to support the project
- Many motivated people from both experiments participate in the development of different features.
Start testing the VMC concept for CBM

Panda decided to join -> FairRoot: same Base package for different experiments

R3B joined

EIC (Electron Ion Collider BNL) EICRoot

SOFIA (Studies On Fission with Aladin)

SHIP - Search for Hidden Particles

First Release of CbmRoot

MPD (NICA) start also using FairRoot

ASYEOS joined (ASYEOSRoot)

GEM-TPC seperated from PANDA branch (FOPIRoot)

CALIFA (CALorimeter for the In Flight detection of γ rays and light charged pArticles)

ENSAR-ROOT Collection of modules used by structural nuclear physyics exp.
FairRoot in nutshell

- An open source project (LGPL V3) available from GitHub
  https://github.com/FairRootGroup/FairRoot
- Simulation-, Reconstruction-, and Analysis-Framework (not only) for the FAIR experiments
- 2003 started as 2 person project for the CBM experiment
- 2014 ≈ 10 experiments use FairRoot as base for their developments
- Core team of 5 Developers (3.5 FTE)
- Many people contribute to make the project a success
Design

• Re use existing software and tools (use standards)
• Code should run on all platforms
• Framework should be
  o Easy to install
  o Easy to use
  o Should allow fast development cycles
  o Flexible to easily change experimental setup
  o Extensible for new developments
Easy to install

• Provide packages with all dependencies (ROOT, Geant3, Geant4, CMake, Boost, ...) plus scripts for automatic installation on all systems

• Use CMake as build system and CTest/CDash for automatic testing and QA

• Works on Mac OSX and many Linux derivatives (Debian, Ubuntu, Suse, Fedora, Scientific Linux), probably on many more which are not tested by us
Easy start for beginners

• Simulation and reconstruction examples are available
• Template for creating new detector setups are delivered with a rename script (Detector classes, data classes, ... etc can be created in seconds)
• Geometry can be defined directly in code, in simple ASCII format, or taken from ROOT files (TGeo Format)
• Simulation results can be displayed and analyzed with plain ROOT
• Tools to visualize the geometry and the tracks are immediately available to the users
Flexibility

• Define run configuration at runtime
  o Use Root macros to define the experimental setup or the tasks for reconstruction/analysis
  o Use Root macros to set the configuration (Geant3, Geant4, ...)
• No executable
  o Use plug-in mechanism from Root to load libraries only when needed
• No fixed simulation engine
  o Use different simulation engine (Geant3, Geant4, ...) with the same user code (VMC)
Flexibility

• No fixed output structure
  o Store only the registered data classes to file
  o Use a dynamic event structure based on Root TFolder and TTree which is created automatically
  o Data output possible after each step
  o Different data levels can be connected via “Friend mechanism” in ROOT

• Simulation and reconstruction can be done in one go or in several steps

• Parameter handling
  o Use the parameter manager developed for the HADES experiment
  o Decouple parameter handling in FairRoot from parameter storage
  o runtime data base IO to/from
    • ASCII files
    • Root files
    • Database
Simulation:

• The framework deliver a set of base classes, which has to be specialized by the user to describe his detector. i.e:
  - Detector
  - Module
  - MCPoints
  - Magnetic field
  - ...etc

• The IO is handled completely by the framework

• Simulation is steered and configured via root macros
Radiation length Manager in FairRoot
FairRadLenManager

Example: Contributions of different functional parts of the MVD to the overall material budget
FairRadMapManager:
What energy dose will be accumulated during a certain time of operation?

- Create all physical volumes with correct material assignment
- Run the simulation engine
- FairRadMapManager will sum up every deposited energy in each volume in the geometry
FairRadGridManager:
What dose rate is expected at a certain space point/region?

Determine the particle fluence through a certain boundary (surface) and deduce a map. Knowing the volume and density of the object of interest and the specific energy loss doses can be estimated.
Real Data – run with cosmic (NeuLAND at R3B)
Beam run at R3B - Neutrons

Neutron peak: 23 cm/ns – close to beam velocity
Gamma peak: close to 30 cm/ns
Finite resolution due to many different sources of background – wrong length assumption

Charge distribution after velocity cut:
Simulation vs. Data Comparison

Energy deposit per module, beam energy range from 255 – 1500 AMeV
Fast Simulation

• Fast Simulation reads same VMC stack to get particles, therefore all event generators are supported with no changes to be done to the Fast Sim codes;

• Fast Simulation may use acceptance parametization calculated from Full Simulation or fast helix approximation for charged particles;

• Fast Simulation also works as a converter from ASCII to ROOT for event generators
Fast Simulation

• The same application, just different configuration:
  – Event generators just push the event into the stack, no transport is taking place
  – Detector response is presented as FairTasks (TTask)
  – The output has the same form as full simulation
Fast Simulation: Concept

Full Simulation  Fast Simulation

Event Generation

Particle Transport
Digitization
Calibration
Reconstruction

Effective parametrization
- acceptance cuts
- resolution smearing
- PID info

Physics Analysis
Testing and building system

• Cmake
  - Create Makefiles (and/or project files) for different platforms.
  - Test support.
  - Large user base assures support.

• CDash to handle data created with CMake
  - PHP framework
  - MySQL database

• Both tools are open source.
Event Display based on ROOT EVE package

Runs out of the box for track visualization in simulation
The ALFA project

A common concurrency framework for ALICE and FAIR experiments

How to distribute the processes?
How to manage the data flow?
How to recover processes when they crash?
How to monitor the whole system?

......
ALICE and FAIR: Why?

- A common framework will be beneficial for the FAIR experiments since it will be tested with real data and existing detectors before the start of the FAIR facility.
  - E.g.: Concepts for online calibrations and alignment can be tested in a real environment, similar to that of the planned FAIR experiments.

- ALICE will benefit from the work already performed by the FairRoot team concerning already implemented features (e.g. the continuous read-out, building and testing system, etc)
How is it with ALFA and FairRoot?

**ALFA**
- AliRoot6
- CbmRoot
- PandaRoot

**FairRoot**
- ROOT
- Geant3
- Geant4
- Genat4_VMC

**Libraries and Tools**
- Protocol Buffers
- BOOST
- ROOT
- ZeroMQ
- CMake
- Geant4
- Genat4_VMC
- Geant3
- VGM
ALFA in a nutshell

• Works for online/offline as well as for simulation.
• Message queue based design.
• Modular design with user codes as plug-in
• Generic interface to transport layer
• Effective use of all available resources
• Can be deployed on a laptop, few PCs, cluster or a existing cloud system.
• Easy configuration, management and monitoring tools
ALFA/FairRoot will use Multi-processing and Multi-threading

• Multi-process concept with message queues for data exchange
  ○ Each "Task" is a separate process, which can be also multithreaded, and the data exchange between the different tasks is done via messages.

  ○ Different topologies of tasks that can be adapted to the problem itself, and the hardware capabilities.

Try to find the correct balance between reliability and performance
Summary

• The Framework started with two men in 2003 as a test for CBM has became the software for the whole FAIR project and many other collaborations

• More than 200 developer from the different experiments are contributing to the experiments code and of course from time to time to the core

• Development of general interest usually finds its way to other experiments by moving from the specific experiment implementation to FairRoot:
  o CAD TO ROOT converter
  o Event Display
  o Geane track propagator
  o Monte-Carlo validation package
  o Event generators

  http://fairroot.gsi.de
More about R3BRoot

fairroot.gsi.de
forum.gsi.de
R3BRoot on freedcamp.com
r3b-sim gsi.de

The FairRoot framework is fully based on the ROOT system. The user can create simulated data and/or supported engines. The analysis tasks in the interface for reading...

Recent blog posts
- New stable version of R3BRoot is available "v14"
- New tag release of FairRoot is available v13.12
- New external packages (apr 13) available
- Service Interruption of the Web/ Subversion and Forum
- New external packages (apr 13) available
- New structure of FairRoot repository
- New external packages (sep 12) available
- PROOF available
- New Forum for FairRoot related discussions

Don't tell me you climbed all the way up here before trying Google??

26/11/14
Discussion and backup
If someone experiments with new features in his local working copy and wants to test them (experimental build)

1. Update (optional)

2. Configure, build and test on local machine

3. Send results automatically to central web page

4. Dashboard prepares and display results

5. Developer check results
If new code enters the central code base (continuous build)

1. Developer commit code

2. Basic checks: Style, etc

3. SVN triggers test server

4. Configure, build and test on local machine

5. Send results automatically to central web page

6. Dashboard prepares and display results

7. In case of problems Dashboard sends an E-mail to Developer and Administrator

Central SVN repository

Dedicated test server

Pass

Fail

Reject commit
From time to time a full check on all supported platforms should be done (nightly build)

1. Update

2. Send results automatically to central web page

3. Dashboard prepares and display results

4. In case of problems Dashboard sends an E-mail to Developer and Administrator

5. In the morning Developers and Administrators check their mails and the dashboard. And the development cycle starts again
Version Management

The Query process
1. Context (Timestamp, Detector, Version) is the primary key
2. Context converted to unique SeqNo
3. SeqNo used as keys to access all rows in main table
4. System gives user access of all such rows
Integrating the existing software:

ROOT Files, Lmd Files, Remote event server, ...

FairRootManager

FairRunAna

Root (Event loop)

FairTasks
- Init()
- Re-Init()
- Exec()
- Finish()

FairMQProcessorTask
- Init()
- Re-Init()
- Exec()
- Finish()