Offline Software Management of the AMS experiment

V. Choutko[1], A. Egorov[1], A. Eline[1], B. Shan[2]

[1]Massachusetts Institute of Technology [2]Beihang University

Abstract

The Alpha Magnetic Spectrometer (AMS) is a particle physics experiment installed and operating on board of the International Space Station (ISS) from May 2011 and expected to last through 2024 and beyond. The AMS offline software is used for data reconstruction, Monte-Carlo simulation and physics analysis. This paper presents how we manage the offline software, including the version control, the building, the documentation, the functional and performance testing, etc.

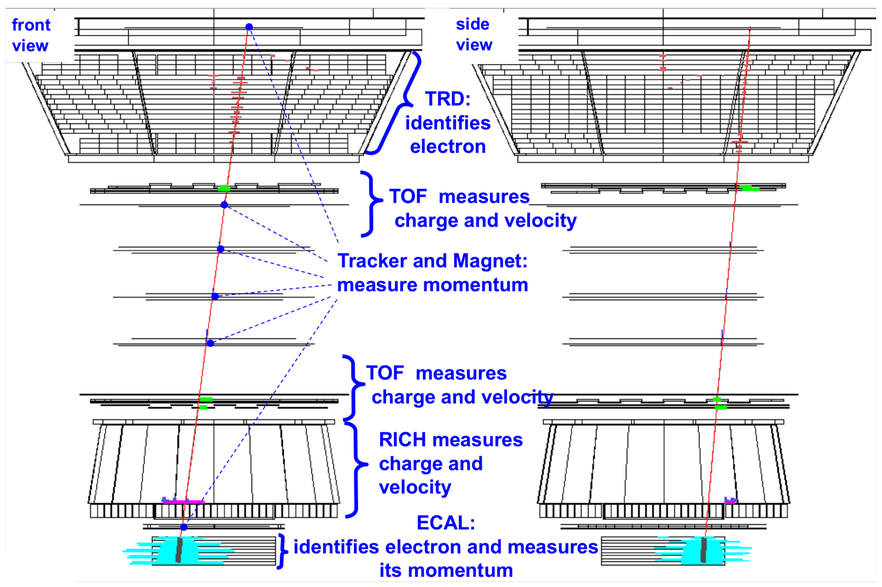
As raw files are of raw events, physics analysis cannot be performed directly on them, instead, reconstruction is required to convert the original detector read outs to physical arguments.

AMS Offline Software

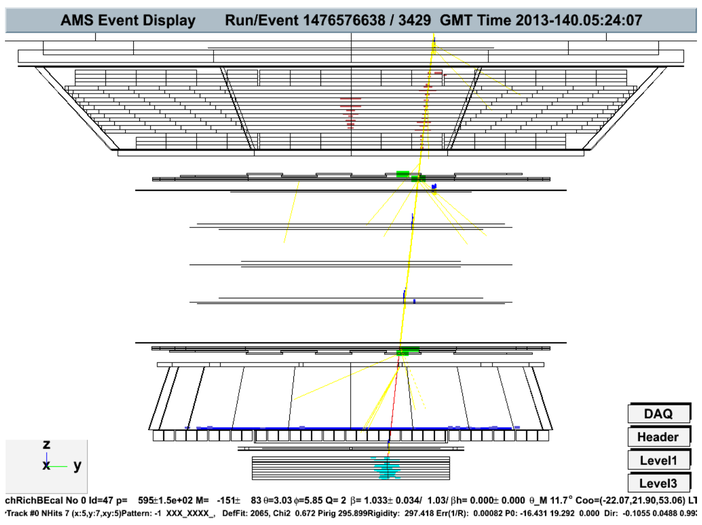
The AMS physics data format is based on CERN ROOT package. Each reconstructed AMS event is represented by ROOT tree object and contains the event header, with general information of the event (around 100 bytes), 8 bytes status word and (referenced) arrays (C++ STL vectors) of reconstructed objects like particle(s), track(s), cluster(s), etc. parameters (typically of 8 kBytes total size). On top of that, for every run the time based information is available, such as average geomagnetic cutoff, detector live time, ISS position parameters, temperatures and voltages of AMS electronics, etc. The timing precision of such data is about 100 milliseconds. The reconstructed event files are grouped together in an event summary file, usually one file per run.

Dedicated AMS offline software is designed to serve both reconstruction and simulation:

Reconstruction: raw events recorded by AMS detector on board of the ISS are converted to reconstructed events, containing all necessary parameters of the particle(s) crossing the detector and being ready for the further physics analysis.



Simulation: Special data cards are given as the input of the offline software, to evaluate AMS detector performance using simulated events, taking into account the cosmic ray fluxes, the detector geometry and all relevant physics processes. The simulation contains two stages, simulating stage uses modified Geant4 library to generate raw events, and reconstruction stage shares exactly the same processing behavior as in the flight data reconstruction.



Version Control

AMS is using Concurrent Versions System (CVS) as the version control system for its offline software.

CVS is a central controlled version control system and the repository (CVSROOT) is located at a shared path where the clients can access. CVSROOT of AMS offline software is located on CERN AFS storage and accessible by all LXPLUS hosts.

Contents: The AMS offline software contains not only the code to build the AMS reconstruction/simulation software, but also the scripts required for the production, for example, the production validation, the production job requesting, etc., and the code of various tools related to AMS offline computing, for example, the "event display", the "fast ROOT reading", "Slow Control Database", etc.

Access Control List (ACL): CVS has a simple access control mechanism relying on the ACL of the file system. AFS uses pts to define the group, so a pts group is defined to allow a group of user to be able to commit changes.

Branches: The reconstruction and simulation software of AMS has several different branches. The main branch "vdev" is the latest version for simulation, and the branch "BXXX_patches" is the stable versions for flight data reconstruction, to keep unique data properties for data analysis.

Migrating to Git

Although CVS is a stable, simple, and efficient version control system, it has quite some limitations/issues compared to the modern competitors. Git a distributed version control system which was designed to be an opposite to CVS, which means, to take CVS as an example of what not to do. Git has a lot more advantages compared to CVS.

Atomic operations: CVS is based on the per-file RCS version control system, so commits (and other operations) are not atomic, which means an interrupted operation may lead to an inconsistent state and a corruption. While in Git all operations are atomic, in this way, the consistency is always assured.

Changesets: Changes in CVS are per file, and they cannot be grouped. This made it very difficult to track the changes of the whole project. Git always refer to the whole project for all commits. This is very important and makes it very easy in to revert or undo whole change and to do partial checkouts.

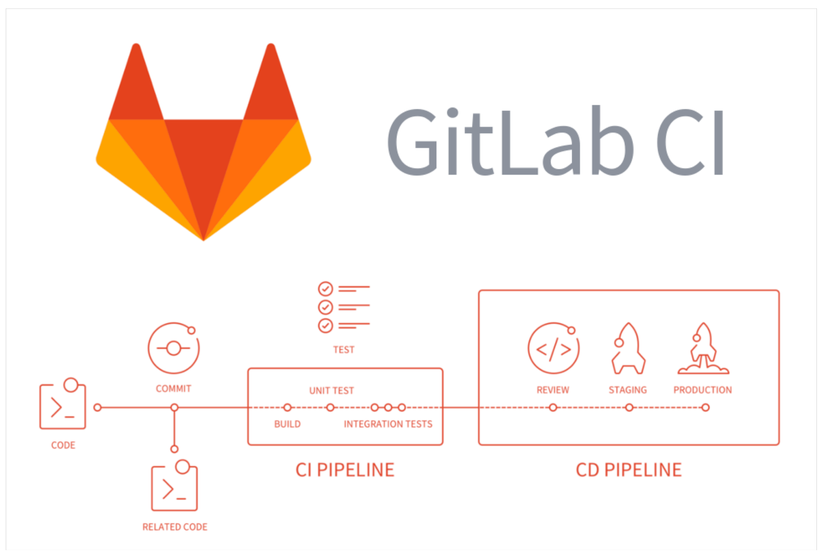
Commit before merge: Since Git is designed to be the opposite of CVS, unlike Git uses merge-before-commit Git uses commit-before-merge. CVS forces users to first update the working directory and resolve conflicts before allowing committing. On the contrast Git allows saving the state first, then merging the changes from others or having other developer resolve conflicts.

On the first stage of the migration is to sync all the CVS contents, including code, history, ACL, branches, etc. to a Git repository and to keep bi-directional synchronization. This means that the changes made in Git will also be applied back to CVS.

After the confirmation of all the editors of the repository, the CVS will be made read only and only to accept the commitments from Git.

Automatic Build, Test, and Deploy

Another advantage of Git is to be able to intergrade automate building, testing, and deploying with the source control, and this will allow the developer to test the commitment.



By hosting at CERN GitLab, we can have the codes to be built automatically once there is any commitment, and the commitment will be rejected if the building fails.