ATLAS trigger simulation with legacy code using virtualization techniques

Gorm Galster[1], **Joerg Stelzer**[2] **and Werner Wiedenmann**[3]

[1] University of Copenhagen, DK
[2] Michigan State University, US
[3] University of Wisconsin, US

E-mail: `gorm.galster@cern.ch, joerg.stelzer@cern.ch, werner.wiedenmann@cern.ch`

Abstract. Several scenarios, both present and future, require re-simulation of the trigger response in the ATLAS experiment at the LHC. While software for the detector response simulation and event reconstruction is allowed to change and improve, the trigger response simulation has to reflect the conditions at which data was taken. This poses a maintenance and data preservation problem. Several strategies have been considered and a proof-of-concept model using virtualization has been developed. While the virtualization with CernVM elegantly solves several aspects of the data preservation problem, the limitations of current methods for contextualization of the virtual machine as well as incompatibilities in the currently used data format introduces new challenges. In this proceeding these challenges, their current solutions and the proof of concept model for precise trigger simulation are discussed.

1. Introduction

Each 50 ns bunches of protons are brought to collide in ATLAS[1], one of the four large detectors experiments at the LHC ring at CERN. At the end of LHC run 1, an average of 35 protons collided per bunch crossing, each generating a spray of particles which are detected by the different sub-detectors of ATLAS.

In order to provide real time data reduction a sophisticated trigger system decide which collision events are stored, effectively reducing the data throughput to hundreds of gigabyte per second. In consequence, all stored data are inherently watermarked by the trigger configuration and trigger algorithms used during data taking. An in-depth analysis of the data thus also becomes a study of the trigger. Having a strategy for simulation of the trigger response is of importance for continued usability of the recorded data.

1.1. ATLAS simulation chain

Physics analyses in high energy physics require that real data are accompanied by simulated Monte Carlo (MC) data. During ATLAS data taking, MC data are produced alongside the real data taking using the same software as is used for online selection and offline reconstruction.

The ATLAS simulation chain is shown in Figure 1. First the detector response to the simulated physics events is simulated. Based on the detector response the trigger simulation adds the trigger decision record to the event before the simulated events are reconstructed and stored.

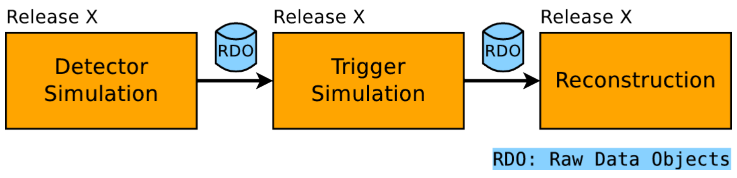

Figure 1. The existing MC production chain. The boxes depict the execution of the respective simulation steps: detector simulation, trigger response simulation and the reconstruction step. The RDO (Raw Data Objects) is the exchange format. The Release X is to denote that the same software version is used for all three steps.

2. Motivation

Several scenarios will require a (re-)simulation of the trigger response:

- An improved description/understanding of the detector response.
- Improved algorithms and methods for offline reconstruction.
- A desire to increase the size of MC samples in future studies.
- The introduction of new event generators.

2.1. Data format challenges

While detector response simulation and event reconstruction should be done with the newest software, the version used for trigger response simulation needs to match that used for data taking. Mismatch between the software versions poses a number of challenges. The most direct challenges are those relating to data format compatibility:

- The trigger response simulation needs to read detector data produced by a new detector simulation. This either implies forward compatibility of the format and content produced by the detector simulation or a possibility to convert the detector response to an older format readable by the old trigger simulation.
- The reconstruction needs to read the old trigger response record. This either implies a conversion step or backward compatibility in the format for the trigger response record such that the reconstruction algorithm can read and understand the produced old trigger response header.

2.2. The environment stability

Besides the challenges related to data formats there are those related to the time scale envisaged for ATLAS. Over a time span of more than a decade it is expected that hardware architectures, operating systems, core components and compilers change, making it impossible to run the old code as-is.

It would thus be necessary to port trigger selection algorithms which, potentially, are not even used any more to the new software releases and certify with every new software release that their selection response remains unchanged. In addition one also has to guarantee that the legacy infrastructure for these trigger algorithms works with the new operating systems, changes in software infrastructure, changes in the compiler and computer hardware developments. Further, knowledge of the working of unused components would have to be preserved so that it is possible to port them continuously to new releases alongside the actual trigger code. This poses a maintenance problem.

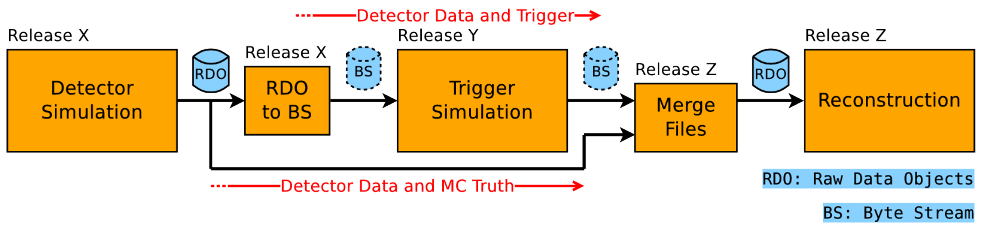

Figure 2. The proposed changes to the existing simulation chain. This chain requires two additional steps and uses an intermediate byte stream (BS) file. The first steps converts the detector specific data of the RDO into BS. After the trigger response record has been added, the RDO is infused with the trigger response record. The resulting RDO is passed on to the reconstruction step.

2.3. Objective and scope
The goal of this study is to investigate viable strategies for a precise trigger simulation reflecting past data taking conditions. The precision requirement can only be realized by the use of the same software as was used during data taking. The proposed strategy should preferably require minimal change to existing infrastructure and additional maintenance even when future developments in hardware and software becomes a reality.

3. Resolving data formats incompatibilities
3.1. The RDO format
The currently used format, in the MC production, as indicated in Figure 1, is the Raw Data Objects (RDO) file format, which is a container format based on ROOT[2] technologies. The RDO files contain as payload structured serialized data objects. The format has the disadvantage with respect to the previously mentioned challenges, that the structure and the content is rapidly evolving in time in response to the demand of more precise detector response simulation and/or addition of new components of the ATLAS detector. While compatibility for reading older files are usually provided, it is not guaranteed, and the complex nature of the format makes it difficult to provide converters for forwards compatibility. Attempts to make such conversions, to allow only a year old trigger simulation to read a modern RDO, were unsuccessful.

3.2. The raw detector format
Another, more native data format exists: the raw detector data format, byte stream[3]. The byte stream format is a chunk-based data format with support for versioning of both the format itself and the data content.

It is already a requirement from ATLAS that the format is kept backwards compatible so that data from any data taking period remain readable. Due to the tight coupling between the byte stream format and thef detector readout, the format is expected to evolve slowly. Further, while not necessarily maintained or kept operational, the code required for the sub-detectors to write old byte stream data already exists. Some sub-detectors of ATLAS already support configuration of the byte stream payload format. Further again, data objects corresponding to detector data are serializable to byte stream (as well as RDO) in order to properly simulate the conditions during online data taking.

The downside of using the byte stream format for simulation is that it inherently only deals with the detector data: MC truth information and other necessary (meta)data are not storable.

3.3. Altering the simulation chain
The trigger response simulation only needs the detector related data of the RDO as input. It adds the trigger response leaving detector data intact. A modified MC production chain, using an intermediate byte stream file, has been developed as shown in Figure 2. Two additional steps are added to the simulation chain. A step that extracts the detector data from the RDO and stores it to the intermediate byte stream file. And the step, after the trigger simulation, which adds the trigger response record to the byte stream file. The trigger response record is read from the byte stream file and added to the RDO. The final RDO file then serves as input to the reconstruction step. These extra steps were implemented within the framework of ATLAS and work has been done to allow for a seamless integration with the existing MC production code. A proof-of-concept setup was rigged and it was shown that the modified simulation chain was viable for all eight tested software versions.

4. Virtualization and data preservation
Virtualization technology solves the before mentioned challenges related to time evolution by allowing effective simulation of older hardware and thus enables a complete encapsulation of operating systems and software.

The use of virtualization, on the other hand, has its down sides: it inherently introduces an overhead. Further, a virtual machine (VM) is for all practical purposes a machine, and not a simple script. Introduction of additional steps running inside of the virtual machine is a major change to the ATLAS MC simulation chain. This implies a major change to the simulation. In addition to the old version of the trigger software, the full machine definitions need to be maintained and kept functional.

Both aspects were addressed in the study using CERN's own VM project, CernVM[4], and its associated technologies.

4.1. The CernVM project
The CernVM philosophy is to have a minimal disk image, which through methods of contextualization, can be configured for use with any version of the experiments software. Disk images are of the order of hundreds of megabytes instead of the several gigabytes of conventional VM disk images. The contextualization information are text files of no more than a few kB. The contextualization and disk inflation then happens when the machine is started.

In order to provide the experimental software, the CernVM images ship with its own HTTP based, locally cache-able, strictly versioned, read-only file system, cvmfs[5]. ATLAS already publishes its software releases and conditions data onto cvmfs file servers.

4.2. Proof of concept implementation
Using libvirt[6], an API for platform virtualization management, with KVM (Linux Kernel-based Virtual Machine)[7] a contextualization was developed building on top of the HEPPIX contextualization method supported by CernVM. The contextualization method, which is mainly intended for grid sites to adjust the raw image to local site policies, was used to tamper enough with the systems initialization process as to make a script that would:

- create and start a fresh CernVM instance,
- contextualize it appropriately,
- have it setup ATLAS release software from cvmfs,
- execute a script with user specified parameters, i.e. the trigger simulation,
- ship out results and log files over xrootd[2],
- and finally dispose of the machine when done.

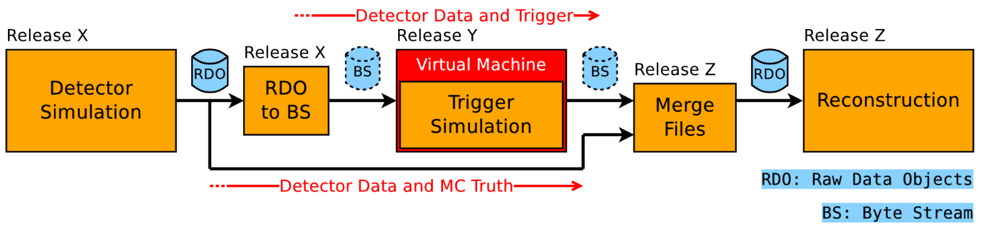

Figure 3. The modified simulation chain where the trigger simulation has been encapsulated in a virtual machine. This setup was used for the proof-of-concept model.

The methods used are not suitable for production since, in the production environment, the software is already being executed on virtual machines. As the chosen infrastructure can not be steered dynamically in the ways described here, this method may lead to a highly undesirable "Russian Doll" where a new VM is created inside a VM. Providing a better solution for dynamic VM allocation would require major changes to existing infrastructure. While not suitable for production the method allowed for rapid integration of virtualization technologies into the existing simulation chain.

At the time of writing it is possible to run the simulation chain depicted in Figure 3 using the proof of concept model on a physical computer. Nevertheless, in order to benefit fully from the CernVM project, better contextualization methods are needed. Requirements for workable methods for setting up a specific ATLAS software release at contextualization as well as elegant ways of providing certificates and credentials for accessing storage services and databases have been fed back to the CernVM project. Further investigation of how to integrate virtualization as seamlessly as possible into the MC production is also needed.

5. Conclusions

It has been demonstrated how to simulate the trigger precisely by using older software versions for the trigger simulation while maintaining modern software versions for detector simulation and reconstruction.

This was achieved by using a more stable data format tightly coupled to the detector hardware. The proposed model introduces only moderate maintenance effort.

Virtualization solutions seem tractable even when software and hardware change radically. The CernVM project already fulfills most of the requirements but more work is needed before production scalability can be assessed.

References

[1] ATLAS Collaboration 2008 The ATLAS experiment at the CERN large hadron collider *JINST* **3** S08003
[2] CERN ROOT Team 2013 ROOT An Object-Oriented Data Analysis Framework `http://root.cern.ch/download/doc/ROOTUsersGuideA4.pdf`
[3] Anjos A d, Beck H P, Gorini B, Vandelli W 2011 The raw event format in the ATLAS Trigger & DAQ *EDMS* 445840
[4] Segal B *et al* 2010 LHC Cloud Computing with CernVM *PoS* **ACAT(2010)** 004
[5] Blomer J *et al* 2011 Distributing LHC application software and conditions databases using the CernVM file system *J. Phys.: Conf. Series* **331** 042003
[6] Red Hat 2013 libvirt: The virtualization API `http://libvirt.org/`
[7] Red Hat 2013 KVM: Kernel Based Virtual Machine `http://linux-kvm.org/`