ATLAS Trigger Simulation with Legacy Code using Virtualization Techniques

A presentation for:

Advanced Computing and Analysis Techniques in Physics
Beijing, May, 2013

by Gorm Galster

in collaboration with
Joerg Stelzer and Werner Wiedenmann

Outline

Physics Motivation

ATLAS Simulation Chain
- Data Format Challanges
- Proposed Changes

Virtualization for Running Legacy Code
- Virtualization with CernVM
- Status and Requirements

Lessons Learned

Motivation for Running Trigger Simulation

Several scenarios will require a re simulation of the trigger response, i.e.

- Improved description of the detector response.
- Improved offline reconstruction.
- Increasing the size of MC samples in future studies.
- Introducing new physics triggers.
- MC production don't store trigger result - starts from scratch and needs resimulation of the trigger.
- Introduction and test of new event generator.

The Existing Simulation Chain

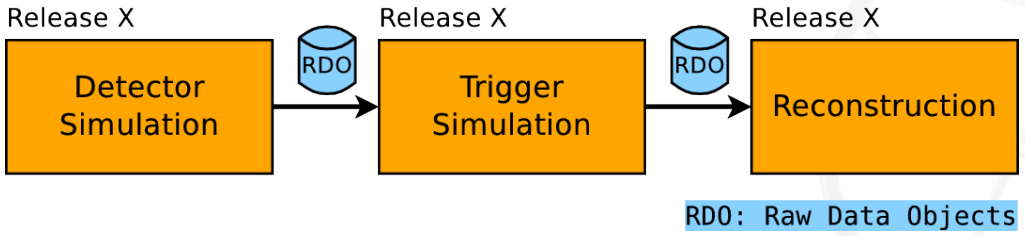

Figure 1: Depiction of ATLAS simulation chain. Generated events are put through Detector Simulation, Trigger Simulation and later Reconstructed. A single release is used for all steps.

- Reconstruction and Detector Simulation should be done with the newest software.
- Trigger Simulation software need to match that used for data taking - precise simulation and limited maintenance.

Aim

The goal is to run the trigger simulation reflecting the past conditions at which data was taken. This implies:

- Finding a viable way to run the trigger simulation with an old release.
- Finding a way, using virtualization technology, to run the trigger simulation when future developments in hardware and software becomes a challenge.
- Preferably, the proposed strategy should require minimal change and additional maintenance.

By using old releases we don't need to maintain legacy trigger code in new releases!

Technologies and payload

The current data format is RDO:

- Container format based on ROOT technology.
- Internal data structure (payload) format is community dictated and rapidly changing.

For retrospective trigger simulation reading and writing of old data formats becomes necessary:

- Data from Detector Simulation needs to be readable by Trigger Simulation (Forward compatibility).
- Data from Trigger Simulation needs to be readable by Reconstruction (Backwards compatibility).

Byte Stream - ATLAS Detector Format

The byte stream format is the native detector data format of ATLAS and has the following relevant attributes:

- Container format changes *very* slowly.
- Payload format is simple.
- Payload format changes slowly - tightly coupled with detector readout.
- Backwards compatibility is guaranteed.
- Forward compatibility is easier to provide - simpler format, slower changes.

Byte Stream as Intermediate File Format

As an intermediate file for the simulation chain it is important to notice for byte stream, that:

- It supports all raw data necessary for running the trigger simulation (input for trigger sim.).
- It supports the trigger response record (output of trigger sim.).
- Detector payload version can be controlled. Requirement for providing forward compatibility.
- It is a natural choice - Trigger selction code normally runs on byte stream data.

.. but does NOT support simulation specific data.

Modified Simulation Chain

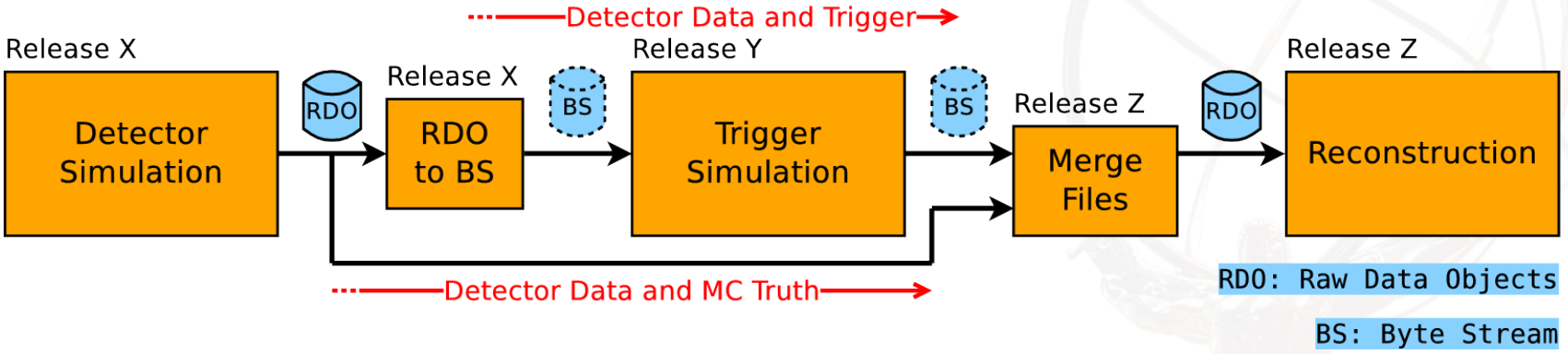

Figure 2: Schematic of modified simulation chain for running old Trigger Simulation. This chain requires two additional steps and uses an intermediate byte stream (BS) file.

Pros and Cons

- This simulation chain solves the legacy data format challenges, provided new releases can write old detector payload.
- The proposed solution does not account for other external storage services which might become upgraded or obsolete.
  - Oracle DB.
  - Geometry data.
  - Conditions data.

Implementation and Status

- Proof of concept implemented as 5 standalone jobs.
- Work is being done to merge with ATLAS production scripts.
  - Release change is natively supported by newest infrastructure.
- Versioning of byte stream, ensuring readability in old release, already in place for some subdetectors

Motivation for Virtualization

Over a period of time we can expect that:

- New hardware technologies are introduced.
- Operating systems change.
- Compiler and core component change.

How to preserve the entire environment? Virtualization seems feasible!

Virtualization as Solution

- Elegantly solves previous problems by abstracting hardware and encapsulating software environment.

- Introduces computational and resource overhead.
- Requires changes to simulation chain - it is a virtual machine, not a script.

The CernVM project in brief

CernVM[1] is a good choice since:

- It is maintained by CERN.
- Machine definitions are small.
- Ships with own version controlled file system, cvmfs.
- ATLAS already distributes releases and conditions data onto cvmfs for grid productions.

The project was presented in more detail at ACAT '08 and '10.

[1] Project web page: `http://cernvm.cern.ch/`

Simulation Chain with Virtualization

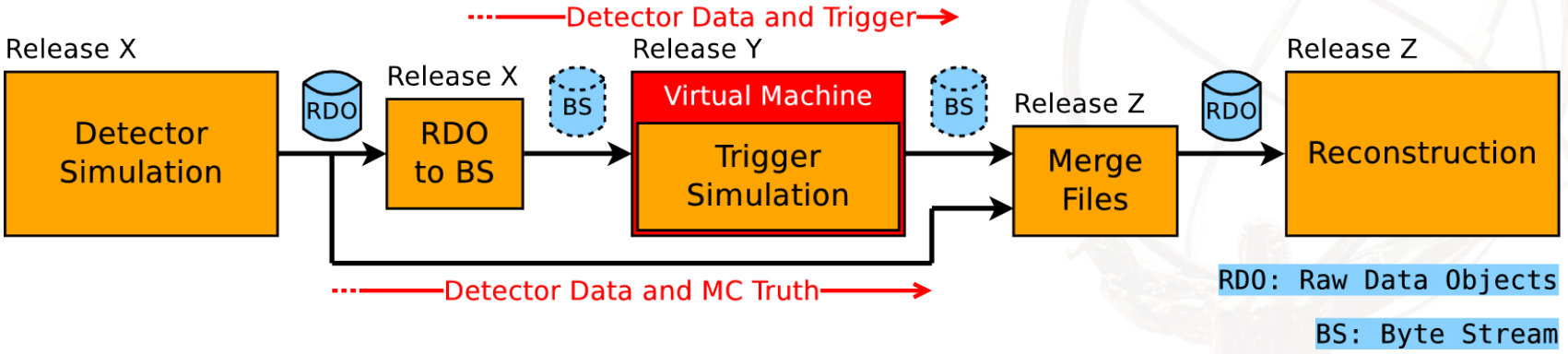

Figure 3: Schematic of the proposed simulation chain shown earlier. Here the trigger response simulation has been encapsulated in a virtual machine.

What is Contextualization

Contextualization is the process of configuring the virtual machine (usually the disk image) for the use in a specific context.

- The contextualization information itself is small and can be stored in small textfiles or databases.
- Several API/syntaxes exists for contexualization - de-facto standard is Amazon EC2.

Contextualization with CernVM

- Several methods for EC2 contextualization are provided by CernVM.
- Currently limited to configuring
  - installed system packages.
  - users and groups.
  - toggle common services.
- Limited possibilities for automatic configuration beyond EC2.

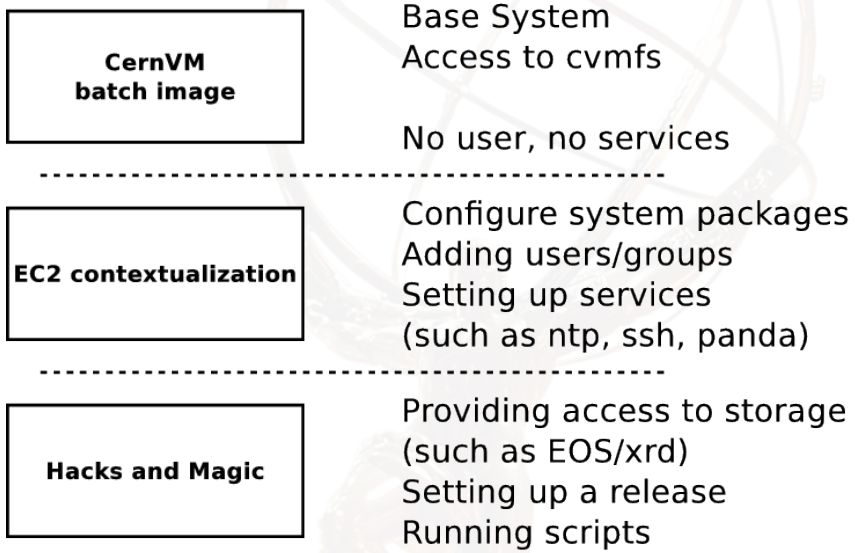

Figure 4: The 3 steps in a full contextualization.

Hacks and Magic

.. or forcefully adding the needed functionality on top of EC2 context:

- Access to storage services and local files - means for input/output.
- Transferring of certificates, credentials etc. to the instance - providing authentication and authorization.
- Setting up a software release - substituting the environment.
- Setting up and disposing of the machine instance - orchestrating.
- Work is being done with CernVM community to find more tractable solutions.

Implementation and Status

A proof of concept script was created using the hypervisor API libvirt with Linux native hypervisor, KVM, as underlying infrastructure.

The script would:

- create and start a fresh CernVM instance.
- contextualize it appropriately.
- have it setup ATLAS release software from cvmfs.
- execute a script - the trigger simulation.
- dispose of the machine when done - shipping out the result over xrootd.

Known Caveats

Some of the design choices for the proof-of-concept are unfit for production environment, especially:

- The choice of contextualization method.
- Handling of input/output files.
- A 'Russian Doll' scenario where a VM is run in a VM could occur if e.g. batch system runs in a VM.
  This is highly intractable.

Requirements for Future Development

To benefit from the virtualization technology we need:

- Better contextualization options - Implement EC2 contextualization for ATLAS.
- Better access to storage and services - Better integration especially with storage services on production farms.
- To go from test bench to production environment - How does it scale?

In conclusion

- It was possible to simulate the trigger precisely by using older software for the trigger simulation while maintaining modern releases for Detector Simulation and Reconstruction.
- The proposed model introduces only little extra maintenance.
- This was achieved by using a more stable data format tightly coupled to the detector hardware.
- Virtualization solutions seem tractable when software/hardware changes radically.
  - The CernVM project already satisfy most needs.
  - More work is needed before production scalability can be assessed.

Thank you for your time! Any questions?