Legacy trigger developments for 2015

The trigger and swimming groups[1]

Brief:

This document proposes and outlines four steps to simplify maintaining and running a legacy version of the software trigger as required for data preservation, re-running for physics reasons, and so-called "swimming": a variant of running the trigger to evaluate decay-time-dependent efficiencies.

The software trigger is controlled by Moore, a Gaudi project and Gaudi application within LHCb. However, other software component projects such as PHYS, HLT, and other applications such as Noether are touched upon in this work plan.

Summary:

Four steps are identified as follows.

1) Repackaging: sub-dividing the swimming code between a Cpp-library in PHYS and a python library in DBASE
2) Modified workflow: preprocessing DST input to Moore to remove additional event-model classes, and postprocessing to recombine the results.
3) Dedicated event model class: using a bespoke class to pass information about particles and hits for swimming in Moore.
4) The complete MDF approach: using a bespoke raw bank instead of a bespoke class (optional).

Each subsequent step is expected to reduce the number of required Moore patches by a factor of around two. If all steps are accomplished we are confident that no significant patching of legacy versions would ever again be required to re-run the trigger software.

The problem:

The number of Moore patch releases to support re-running of the HLT is increasing much higher than linearly due to the increasing number of: Moore Versions to fix, fixes to be applied, and additional swimming-specific developments/problems. This is an unmaintainable situation which needs addressing.

In 2012 we ran essentially only five different versions of Moore for the HLT, the majority of which shipped with almost identical code, such that there were only two real version groups. However, Stripping of our data occurred later with a much more advanced version of the software. Even to read in fully-stripped DSTs, each Moore which ran on real data must be patched based on the gradually increasing number of backwards-incompatible changes in the latest software. Each additional feature only serves to reduce the similarity between the legacy trigger and the trigger which actually ran on real data in the event filter farm.

The two logical Moore sets are still perfectly runnable as they ran in the pit; however, for offline running 17 different patch versions have already been released, and an additional 8 releases of patched component libraries in 2013 to accompany them. Most patch releases required several backported features, and each backport was a complicated exercise in software archaeology, requiring careful discussion, design and implementation by a team of software experts. The median release time for each patch release was two weeks (at 0.5 FTE).

[1] Contact authors: Rob.Lambert@cern.ch and Vladimir.Gligorov@cern.ch

Step 1: Repackaging: sub-dividing the swimming code

- **Description:**
  - repackage existing code so that it is more flexible with the help of DBASE
  - DBASE is a collection of database-like projects in LHCb software. Packages within are special in that they can be chosen at run time, and are not pinned to projects

- **Advantages:**
  - reduces the number of patch Moore/DaVinci releases by approximately a factor of 2
  - reduces the lead time in testing/releasing swimming
  - allows for much greater flexibility in the swimming python
  - allows for closer management of the python direct by the swimming co-ordinators

- **Requirements:**
  - make Phys/Swimming obsolete on the head
  - move the C++ code into a newly named Phys/SwimmingCpp package (or similar)
  - move the python code into a newly named SwimmingConf package in DBASE
  - rename the configurable to SwimmingConf
  - modify ProdConf to try first the new configurable
  - modify scripts to say "from SwimmingConf import SwimmingConf" instead of "from Configurables import Swimming"

- **Why do we need these steps?**
  - the way we package software in DBASE is slightly convoluted, and in order to avoid accidentally picking up the wrong configurable from the wrong place, we do need to split the existing package, and rename the configurable. So, we get some significant advantages from this step, but it doesn't come for free.

- **Required Effort:**
  - Code Migration: Rob Lambert
  - ProdConf Changes: Marco Clemencic
  - Modified swimming scripts and testing/prototyping: the swimming team.

- **Timescale:**
  - February 2014

- **Remarks:** this is a complete prerequisite for all subsequent steps.

Step 2: Modified workflow: Pre-processing DST input to Moore

- **Description:**
  - Modify the workflow, so that the job of Moore is much simplified.
  - a) Preprocessor step, prepare a file with only the raw event plus any particles/vertices/tracks/hits needed to swim
  - b) pass this intermediate file to Moore,
  - c) Postprocess: merge back in the swimming reports to the original DST

- **Advantages:**
  - Only the most recent software needs to understand new developments
  - reduces the number of required Moore patches by a large factor
  - runs a much more similar version of Moore in production as was run in the pit
  - enhances software backwards compatibility and data preservation

- **Requirements:**
  - Custom micro-DST + raw event "writer", possibly in the Noether project
  - Novel generic event-wize merger of Root-like DST-like files.

- **Why do we need these changes:**
  - In order to split the workflow and intelligently combine the two outputs.

- **Required Effort:**
  - Custom writer: A member of the swimming team, with the help of Chris Jones.
  - Novel merger: Can only be done by Marco Clemencic (he's nominally agreed to do it)
  - testing/prototyping: A member of the swimming team

- **Timescale:**
  - Approximately June 2014

- **Remarks:**
  - This step does not protect us against changes in persistency, or redefinition of particles/vertices themselves (as has happened in the past), for that, step 3 is required.
  - Step 1 and 2 are useful for generic re-running of the trigger, steps 3 and 4 are more swimming-specific

Step 3: Dedicated event model class

- **Description:**
  - Modify the required input for the swimming, so that swimming does not require access to particles and vertices from stripped DSTs. Instead use a much more stable interface class directly for the purpose, with only that information which is needed.

- **Advantages:**
  - never have to patch Moore to get around issues of Particle/Vertex

- **Requirements:**
  - create a bespoke class with a four-vector, hits, and a link to the original particle probably using a string for the location and the key within the particle container.
  - significantly change the swimming to use this bespoke class instead of a particle
  - Create an interpreter algorithm, which can understand and re-link the new swimming reports to the original particles during post-processing.

- **Why do we need these changes:**
  - We think it is very likely that LHCb event model classes, required: vertices, hits, particles, and containers of those C++ objects; will change in the future. Such a change might occur given the new flexibility afforded by C++11, and may be a natural consequence of Root6.

- **Required Effort:**
  - The swimming team

- **Timescale:**
  - 2015 start-up.
  - Providing that the Moore versions which run in the pit in 2015 have this event model class, algorithms and the workflow which uses this class can be developed later.

- **Remarks:**
  - This protects us against changes in the classes currently used for swimming, but does not protect us from generic changes in persistency. For such generic changes, we would require step 4 (optional but very useful).

Step 4: The complete MDF approach:

- **Description:**
  - Step 3 still requires us to pass some Root-format file into Moore, however, Moore when it ran on the real data did so directly from MDF-formatted raw banks
  - By encoding the bespoke class of step 3 into a raw bank, we could pass an MDF-formatted file to Moore from the pre-processing step.

- **Advantages:**
  - Running Moore on an MDF file means never worrying about C++ or persistency changes ever again.
  - The persistency system for ROOT files would be free to change in the future.
  - No patching would ever be required to backport new software features to Moore.
  - Patches to modify the raw bank or swimming code itself might be required, but these can be very easily ring-fenced into specific packages and need no code archology.

- **Requirements:**
  - raw bank definition, added to Enum
  - a decoder and encoder for this new raw bank

- **Why do we need these changes:**
  - It would be so much nicer never to need a patch of Moore ever again.

- **Required Effort:**
  - same as step 3

- **Timescale:**
  - same as step 3

- **Remarks:**
  - We need not do both step 3 and 4, we could combine them, and aim straight for a raw bank rather than an intermediate bespoke class in order to perform the swimming
  - Step 4 is optional, it is likely that step 3 alone will solve the majority of patch requirements. Where step 3 is very much necessary, step 4 is a logical extension of it.

Conclusion

We have proposed four steps for LHCb toward a patch-free system of re-running legacy high-level trigger software in the future.

If all four steps are performed and ring-fenced within dedicated packages, we can consider releasing much simplified patch versions of 2011 and 2012 Moore, in early 2015, which themselves would then be immune to future software developments.

The first two steps are useful in their own right as mechanisms for data preservation, enabling any past version of Moore to be re-run as it was in the pit. The second two steps are more swimming-specific, to ensure that running the swimming requires at most a set of well-defined ring-fenced packages to be added to any legacy Moore version. Step four would be the ultimate step in this procedure, but it is noted to be optional, it is likely that even with the first three steps the maintenance load would become manageable once more.

Although significant effort is required to implement the proposed changes, it is already much less effort that it would take to, for example, release a patched Moore from 2011 should any new Root or Gaudi features be required, this proposal reduces the maintenance load of the trigger software by a huge fraction. In 2013 fully half of all releases of trigger software were to support legacy code.

If this new system is put in place by 2015 we can be confident to re-run trigger software for the many years of LHC running to come, with minimal software interventions.