LHCb: current model and future requirements

M. Clemencic, B. Couturier *on behalf of LHCb*

May 30, 2018

CERN - LHCb

Table of contents

1. Current LHCb Build system
2. Feedback on the current system
3. Requirements

Current LHCb Build system

Main characteristics

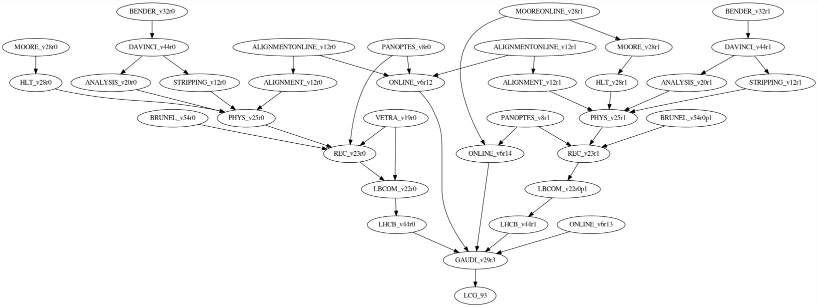
- Gaudi based (`https://gaudi.web.cern.ch/gaudi/Gaudi`)
  CMake build, using a toolchain to locate dependencies
- External dependencies taken from the LCG stack
  Compiled using LCGCMake
- Uses the LCG_XX.txt file to locate externals
- Deep stack with O(30) applications and common libraries (O(6e6 LOC))
- Packaged as *custom* RPMs with dependencies to the LCG RPMs
- Environment configured using lb-run (and lb-dev) custom tools
  lb-run allows ”layered” configuration (e.g. continuous integration on top of production repository)

Platforms and compilers

- Systems: **SLC5, SLC6, CentOS7** depending on stacks
  interested in Ubuntu for smaller scale tests
- Compilers
  - **gcc62** for 2018 stack
  - **gcc7, gcc8** for Run 3 stack
  - also interested in clang (potentially the Intel compiler too)
- Using Python 2.7, planning the migration to Python 3

HepOSlibs RPM required on top of base system or SLC/Centos

LCG 93 / Gaudi v29r3 stack



Continuous integration

Uses OpenShift hosted **Jenkins** instance

- Builds performed on **OpenStack VMs**
- **Docker** used to deal with multiple systems (SLC5, SLC6, CentOS7)
- Builds based on **Released LCG + LCG nightly dev3 and dev4**
- Copied to **CVMFS** (/cvmfs/lhcbdev.cern.ch)

Installation

Installations using the *lbinstall* tool, as a user

- Production RPMs produced by Jenkins, copied to **EOS backed YUM repository**
- LCG RPMs taken directly from the **EP-SFT YUM repository**

Current installations:

- Released on **CVMFS** (/cvmfs/lhcb.cern.ch)
- A number of **local installations** throughout the collaboration e.g. Trigger LiveCD image used for tenders
- **LHCb online setup (trigger) uses CVMFS**

Post install script used to relocate installed files.

Feedback on the current system

Use within LHCb

- LCG stack's very controlled aspect **works well for current production software** (reconstruction, simulations, trigger)
  - Generators are more difficult to deal with as they evolve separately
- LHCb Production releases are **not dynamic/flexible enough for Analysis**
  - installing latest version of Python modules on top is not very easy
  - Some analysts have their own stacks (conda...)
- Our Grid **middleware** does not easily fit in the framework and has to be **prepared separately**
  - We even have incompatibilities between the dependencies (e.g. Boost for GFAL2)
- Would be nice to have **finer granularity in some packages** e.g. for ROOT, c.f. Debian release example
- **Running/Rebuilding old stacks can be problematic** (c.f. later)

History and Preservation

- Need to keep running **CMT** built software (trigger, simulation) from **2010 onwards**
- May need to **patch/rebuild old versions of the simulation and trigger applications**
- Not easy to rebuild part of the stack long after initial release to update versions of some externals
    - e.g. We need to run the old version of the trigger, but the version of xrootd compiled within LCG is not compatible with the WLCG server.
- Systems libraries used are not checked
  can change with OS updates

Issues

- **Separate build system for LCG and LHCb** complicates matter e.g. port to ARM or PowerPC
- Base platform definition unclear: Need HepOS lib RPM
  - N.B. Even that is not an exhaustive list of packages
- Not easy to distribute the work of integrating new externals
- Difference in release cycles between externals and generators complicates the management of the stack

Some problems but still ***a lot easier and smoother*** *than in the CMT, pre-RPM days*

Requirements

Configuration and Compilation

- Ideally **one tool** for externals + LHCb software
- Need to easily **build other external packages on top of LCG**
  - We should be able to fix/add packages after the initial release
  - Need to be agile and distributed:
    e.g. developers can build and test new external versions and report results...
- **Better definition of base system** (and which libraries/commands) can be used would be useful
  or avoid using them at all.
- Need release and development mode
- Need for exact **reproducibility of the physics stack**
  Evolution of the middleware is an issue

Installation

- Packages need to be installable as user (?)
  What about virtual environments?
- Relocatable packages (?)
- Need to install on a shared filesystem (?) (e.g. c.f. architecture constraints)
- Need to easily **manage** multiple install areas
  - Need to be able to remove packages and their dependencies
  - And to produce ”minimal” installations for specific purposes

Need to rethink the requirements, in view of the evolution of the computing landscape

Environment

- Need for flexible system
  Still limiting the complexity from the user's point of view
- Composing the environment for a specific user (e.g. with local changes) is a must

Conclusion

Conclusion

- Current build/release system fulfills the production needs
  But with some issues…
- The system is not flexible enough, hard to debug and heavy on maintenance
  But a lot better that it used to be
- The computing landscape has evolved
  We should profit from new opportunities
- Willing to evolve, and looking forward to the discussions and developments happening within the HSF