LHCb Software for Run 3

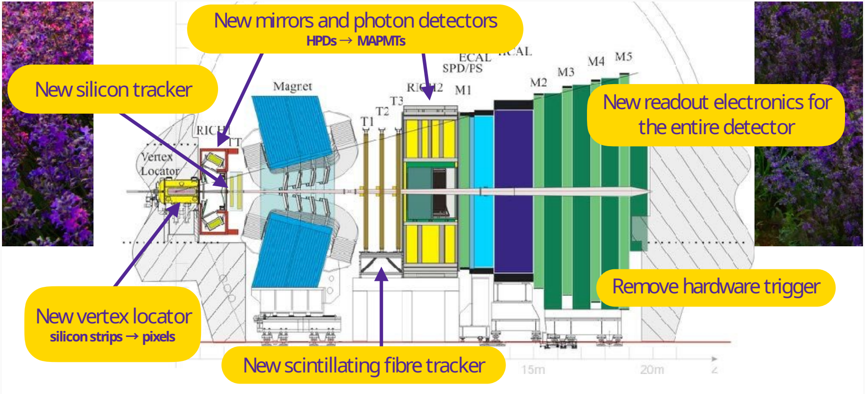
M. Clemencic *on behalf of LHCb Collaboration*

March 5, 2020

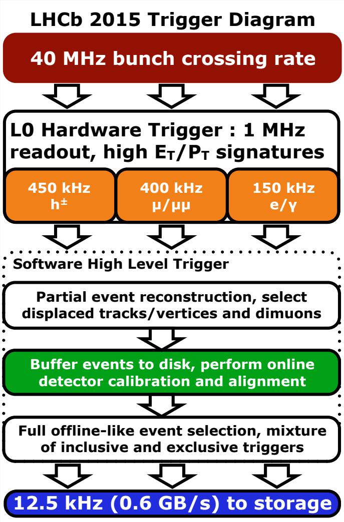
CERN - LHCb

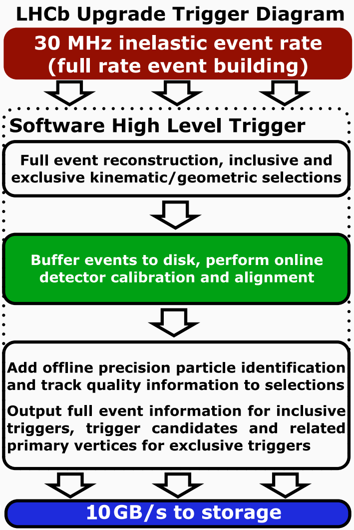
LHCb Run 3

A New Detector



No Hardware Trigger





Software Framework

Go for Multithreading

- Leveraging on the *Gaudi Hive* exercise
  - same framework concepts
  - new interfaces
- *`Gaudi::Functional`*
  - new way of writing Gaudi algorithms
  - function-like signature:

    `Output operator()(const Input1&, const Input2&) const`
  - thread-safe counters (with eventual consistency)
- Processing efficiency
  - small footprint: $\sim$ 20 MB/evt vs. $\sim$ 530 MB/job
  - simple scheduling: $\rightarrow$ 1 thread/event

    (we have small events and small processing time)

Status

- HLT1
  - fully migrated to the new style
  - heavily optimized
- HLT2
  - migration and optimization in progress

GPUs

- *Idea*: run filtering on Event Builder Farm
  - use CPUs for event building and GPUs for filtering
- HLT1 code ported to GPU (project *Allen*)
  - includes a mini framework
- Not yet decided if will use CPUs or GPUs

Software Infrastructure

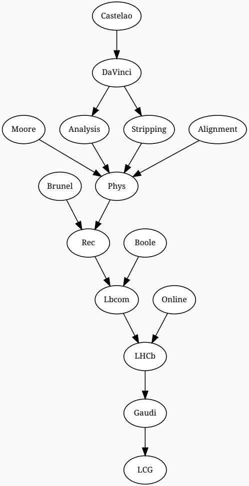
User/Developer Environment

- Runtime and Dev tools decoupled from Physics Software
- Standard Python packages
  - ***pip install LbEnv***
    install the tools for LHCb Software runtime
  - ***pip install LbDevTools***
    for development tools
  - ***pip install LbNightlyTools***
    nightly builds scripts
  - ...
- Shared installation on */cvmfs*
  - currently using Python 2.7 + virtualenv
  - soon to move to Python >= 3.7 + conda

CI/CD and Nightly Builds

- Nightly Build System used for
  - regular integration builds and tests (*nightlies*)
  - on demand integration tests on specific MRs
  - release builds
- OS virtualization via Docker
  - reference images for SLC5, SLC6 and CentOS7
- Refactoring ongoing
  - Jenkins $\rightarrow$???
  - monolithic jobs $\rightarrow$ small interdependent tasks
  - cache and reuse artifacts

Physics Software



- Many projects in a *stack*
  - from the framework Gaudi
  - to analysis application DaVinci
- C++ as the main language
- Python for configuration
- CMake for building
  - refactoring ongoing:
    use standard *modern* CMake
- Custom RPMs for packaging
  - investigating the adoption of Spack
- Shared installation on `/cvmfs`
- Singularity for runtime on Grid resources

External Dependencies

- We rely on the builds by SFT
    - but not everything

AIDA, **Boost**, catboost, chardet, **CLHEP**, coverage, cppgsl, CppUnit, *DD4hep*, doxygen, eigen, fastjet, fftw, flatbuffers, fmt, gdb, graphviz, GSL, **HepMC**, HepPDT, idna, ipython, jemalloc, lapack, libgit2, libunwind, libxml2, lxml, matplotlib, mpmath, networkx, packaging, pathos, pyanalysis, pygraphics, pyqt5, Python, pytools, rangev3, RELAX, **ROOT**, six, sqlite, tbb, tensorflow, Vc, vdt, veccore, vectorclass, wcwidth, XercesC, xgboost, xqilla, xrootd

Detector Description and Conditions Database

- Use DD4hep for detector description
  - need to move away from custom solution
  - migration ongoing
- Git CondDB
  - worked well for Run 2, we keep it for Run 3
  - git bare repositories on dedicated */cvmfs*

Backup/Details

References

- Infrastructure
  - Nightly Builds Modernization: LBCORE-1708
- Gaudi
  - Introduction to Gaudi::Functional
  - Gaudi CMake modernization: gaudi/Gaudi!922 and gaudi/Gaudi!986

HLT1 Throughput

The Hlt1 throughput scaling with a node hosting one AMD EPYC 7702 processor with 64 physical cores and up to 128 logical threads. The dashed line shows an hypothetical linear scaling up to the number of physical cores.