LHCb Framework

- LHCb dataflow reminder
- Tupling

For the LHCb Data Processing and Analysis Project “DPA”

23/09/2020 — FCC–LHCb meeting

Patrick Koppenburg

Nikhef

LHCb Upgrade

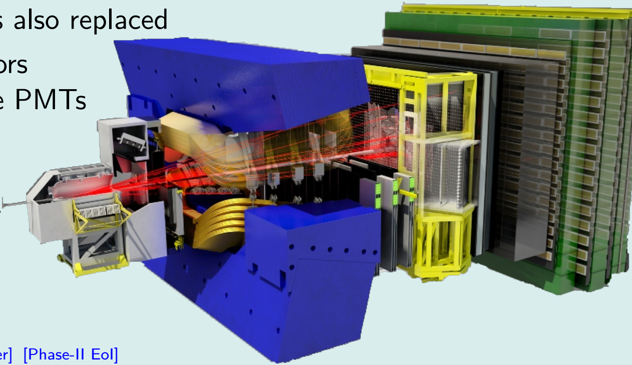
$\mathcal{L} = 2\cdot10^{33}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$ requires some new detectors and 40 MHz read-out clock new electronics

Velo: New pixel vertex detector

Trackers: New scintillating fibre tracker. The upstream tracker is also replaced

PID: Hybrid photodetectors replaced by multi-anode PMTs

➜ 50 fb^{-1} by Run 4.



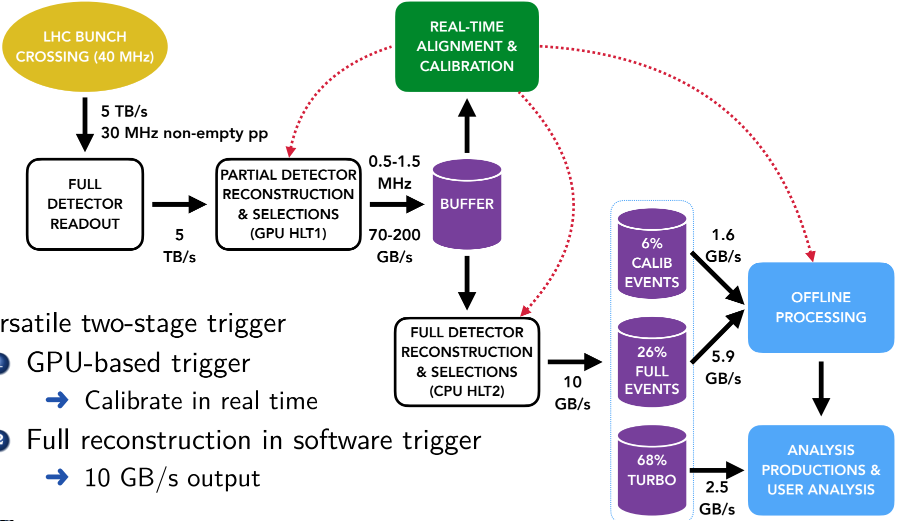
✔ We are preparing another upgrade for Run 5

➜ 300 fb^{-1}

[Upgrade TDR] [Velo] [PID] [Sci-Fi] [Trigger] [Phase-II EoI]

[CERN-LHCC-2014-016]

LHCb Trigger in Run 3


LHC BUNCH CROSSING (40 MHz)
5 TB/s
30 MHz non-empty pp
FULL DETECTOR READOUT
5 TB/s
PARTIAL DETECTOR RECONSTRUCTION & SELECTIONS (GPU HLT1)
0.5-1.5 MHz
70-200 GB/s
BUFFER
REAL-TIME ALIGNMENT & CALIBRATION
FULL DETECTOR RECONSTRUCTION & SELECTIONS (CPU HLT2)
10 GB/s
6% CALIB EVENTS
26% FULL EVENTS
68% TURBO
1.6 GB/s
5.9 GB/s
2.5 GB/s
OFFLINE PROCESSING
ANALYSIS PRODUCTIONS & USER ANALYSIS

Versatile two-stage trigger

1. GPU-based trigger
   - → Calibrate in real time
2. Full reconstruction in software trigger
   - → 10 GB/s output

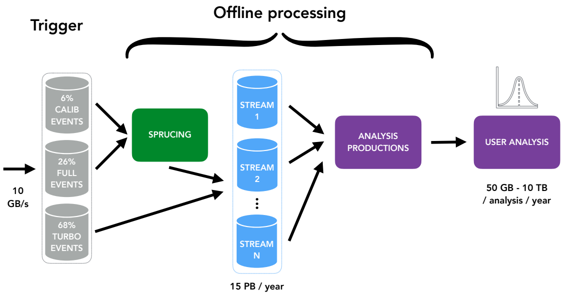
LHCb analysis model for Run 3

The trigger is the final selection. No reprocessing done offline.

Users run on their selected candidates (may add some additional info, when available).

So users just have to say "I want this trigger line and look at these variables"

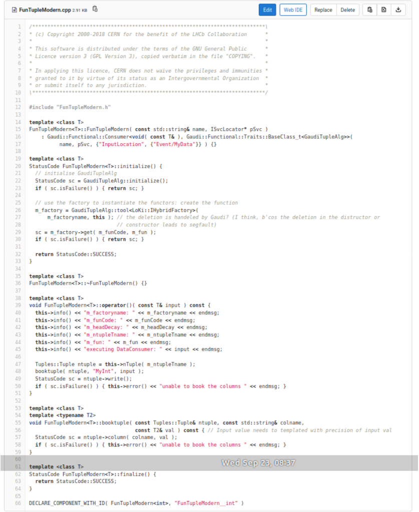
➜ DaVinci application (based on Gaudi)



Tupling

We are rewriting the tuple-filling algorithm. The design of "FunTuple" is

1. It will be based on the same `Gaudi::Functional` base-class as trigger algorithms [DevelopKit]
2. Will load **functors** that produce numbers, `M`, `PT`, `IPCHI2......`
   - Some are complex, like "decay-time of a refitted $B_s^0 \to J/\psi\phi$ candidate with B_s^0 constrained to the PV and B_s^0 and J/ψ mass-constrained."
   - Trigger variables are not stored in data, and we want internal consistency.

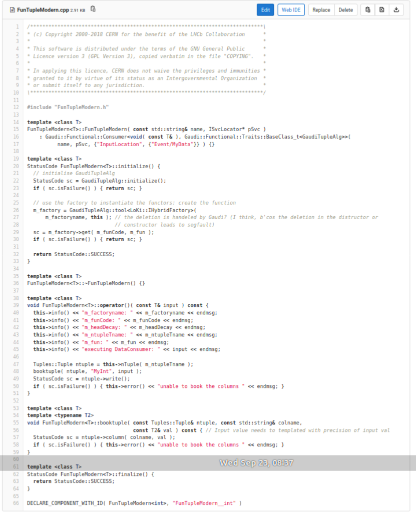


```cpp
/*****************************************************************************\
* (c) Copyright 2000-2018 CERN for the benefit of the LHCb Collaboration      *
*                                                                             *
* This software is distributed under the terms of the GNU General Public      *
* Licence version 3 (GPL Version 3), copied verbatim in the file "COPYING".   *
*                                                                             *
* In applying this licence, CERN does not waive the privileges and immunities *
* granted to it by virtue of its status as an Intergovernmental Organization  *
* or submit itself to any jurisdiction.                                       *
\*****************************************************************************/

#include "FunTupleModern.h"

template <class T>
FunTupleModern<T>::FunTupleModern( const std::string& name, ISvcLocator* pSvc )
    : Gaudi::Functional::Consumer<void( const T& ), Gaudi::Functional::Traits::BaseClass_t<GaudiTupleAlg>>(
          name, pSvc, {"InputLocation", {"Event/MyData"}} ) {}

template <class T>
StatusCode FunTupleModern<T>::initialize() {
  // initialise GaudiTupleAlg
  StatusCode sc = GaudiTupleAlg::initialize();
  if ( sc.isFailure() ) { return sc; }

  // use the factory to instantiate the functors: create the function
  m_factory = GaudiTupleAlg::tool<LoKi::IHybridFactory>(
      m_factoryname, this ); // the deletion is handeled by Gaudi? (I think, b'cos the deletion in the distructor or
                             // constructor leads to segfault)
  sc = m_factory->get( m_funCode, m_fun );
  if ( sc.isFailure() ) { return sc; }

  return StatusCode::SUCCESS;
}

template <class T>
FunTupleModern<T>::~FunTupleModern() {}

template <class T>
void FunTupleModern<T>::operator()( const T& input ) const {
  this->info() << "m_factoryname: " << m_factoryname << endmsg;
  this->info() << "m_funCode: " << m_funCode << endmsg;
  this->info() << "m_headDecay: " << m_headDecay << endmsg;
  this->info() << "m_ntupleTname: " << m_ntupleTname << endmsg;
  this->info() << "m_fun: " << m_fun << endmsg;
  this->info() << "executing DataConsumer: " << input << endmsg;

  Tuples::Tuple ntuple = this->nTuple( m_ntupleTname );
  booktuple( ntuple, "MyInt", input );
  StatusCode sc = ntuple->write();
  if ( sc.isFailure() ) { this->error() << "unable to book the columns " << endmsg; }
}

template <class T>
template <typename T2>
void FunTupleModern<T>::booktuple( const Tuples::Tuple& ntuple, const std::string& colname,
                                   const T2& val ) const { // Input value needs to templated with precision of input val
  StatusCode sc = ntuple->column( colname, val );
  if ( sc.isFailure() ) { this->error() << "unable to book the columns " << endmsg; }
}

template <class T>
StatusCode FunTupleModern<T>::finalize() {
  return StatusCode::SUCCESS;
}

DECLARE_COMPONENT_WITH_ID( FunTupleModern<int>, "FunTupleModern__int" )
```

Tupling

We are rewriting the tuple-filling algorithm. The design of "FunTuple" is

3. These functors are **the same** as those run in the HLT. The χ^2_{IP} in my tuple should be the exact value used in the trigger to select the candidate
    - "Improving" only leads to inefficiency.
4. Developments ongoing on all fronts: Framework, Event Model, Functors, Simulation-Matching *etc.*
    - ✓ New code based on online tools optimised for performance



```cpp
/*****************************************************************************\
* (c) Copyright 2000-2018 CERN for the benefit of the LHCb Collaboration      *
*                                                                             *
* This software is distributed under the terms of the GNU General Public      *
* Licence version 3 (GPL Version 3), copied verbatim in the file "COPYING".   *
*                                                                             *
* In applying this licence, CERN does not waive the privileges and immunities *
* granted to it by virtue of its status as an Intergovernmental Organization  *
* or submit itself to any jurisdiction.                                       *
\*****************************************************************************/

#include "FunTupleModern.h"

template <class T>
FunTupleModern<T>::FunTupleModern( const std::string& name, ISvcLocator* pSvc )
    : Gaudi::Functional::Consumer<void( const T& ), Gaudi::Functional::Traits::BaseClass_t<GaudiTupleAlg>>(
          name, pSvc, {"InputLocation", {"Event/MyData"}} ) {}

template <class T>
StatusCode FunTupleModern<T>::initialize() {
  // initialise GaudiTupleAlg
  StatusCode sc = GaudiTupleAlg::initialize();
  if ( sc.isFailure() ) { return sc; }

  // use the factory to instantiate the functors: create the function
  m_factory = GaudiTupleAlg::tool<LoKi::IHybridFactory>(
      m_factoryname, this ); // the deletion is handeled by Gaudi? (I think, b'cos the deletion in the distructor or
                             // constructor leads to segfault)
  sc = m_factory->get( m_funCode, m_fun );
  if ( sc.isFailure() ) { return sc; }

  return StatusCode::SUCCESS;
}

template <class T>
FunTupleModern<T>::~FunTupleModern() {}

template <class T>
void FunTupleModern<T>::operator()( const T& input ) const {
  this->info() << "m_factoryname: " << m_factoryname << endmsg;
  this->info() << "m_funCode: " << m_funCode << endmsg;
  this->info() << "m_headDecay: " << m_headDecay << endmsg;
  this->info() << "m_ntupleTname: " << m_ntupleTname << endmsg;
  this->info() << "m_fun: " << m_fun << endmsg;
  this->info() << "executing DataConsumer: " << input << endmsg;

  Tuples::Tuple ntuple = this->nTuple( m_ntupleTname );
  booktuple( ntuple, "MyInt", input );
  StatusCode sc = ntuple->write();
  if ( sc.isFailure() ) { this->error() << "unable to book the columns " << endmsg; }
}

template <class T>
template <typename T2>
void FunTupleModern<T>::booktuple( const Tuples::Tuple& ntuple, const std::string& colname,
                                   const T2& val ) const { // Input value needs to templated with precision of input val
  StatusCode sc = ntuple->column( colname, val );
  if ( sc.isFailure() ) { this->error() << "unable to book the columns " << endmsg; }
}

template <class T>
StatusCode FunTupleModern<T>::finalize() {
  return StatusCode::SUCCESS;
}

DECLARE_COMPONENT_WITH_ID( FunTupleModern<int>, "FunTupleModern__int" )
```

Conclusion

We are rewriting everything

- All our software is based on Gaudi
- We welcome external contributions