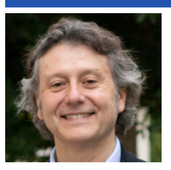
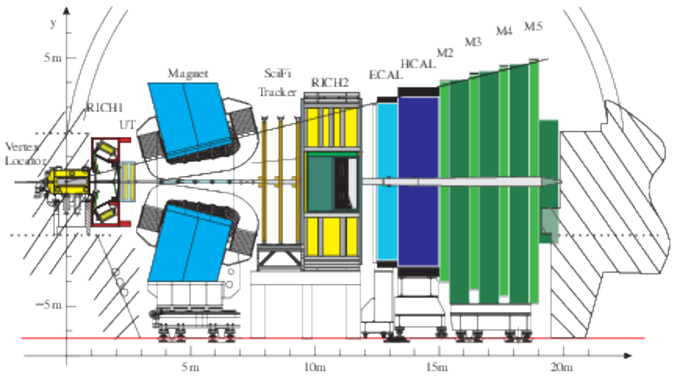
Event and data persistency models for the LHCb Real Time Analysis System

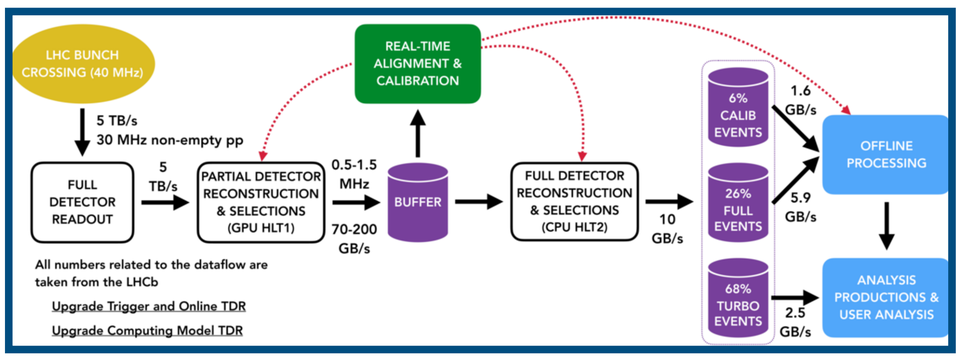


LHCb collaboration - RTA project - Presented by **Xavier Vilasís Cardona**
La Salle, Universitat Ramon Llull

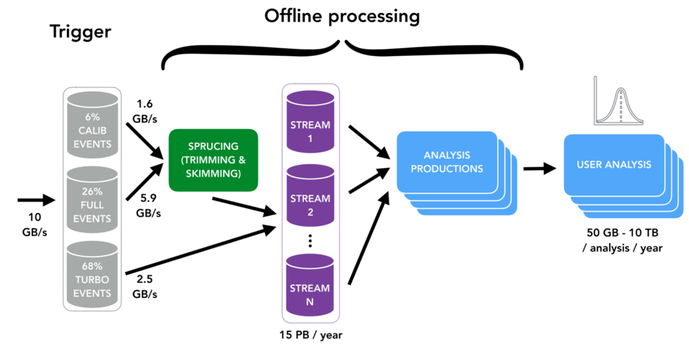
LHCb current detector



CERN-LHCC-2018-014, LHCb-TDR-018



LHCb-FIGURE-2020-016



LHCb-FIGURE-2020-016

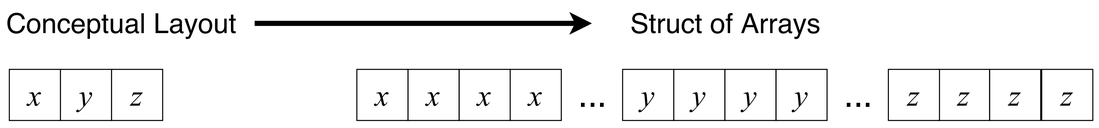
Event Model

C++ classes that represent the data flow from detector raw banks to particles for analysis.

Requirements

Flexible data structures to accommodate vectorisation, multithreading, task-based algorithms. Should be adapted to Single Instruction Multiple Data **SIMD** framework. Currently used for **HLT2**.

SOA : Struct of Arrays - well suited for SIMD approach



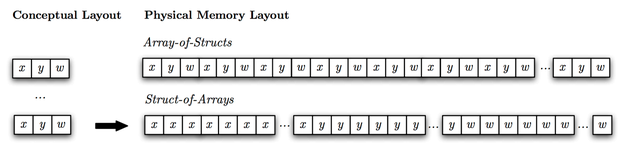
SOACollections : dynamically-resizable collection of SOA

Example : simple track collection

```
// Define tags:
struct Momentum : float_field {};
struct Index : int_field {};
struct LHCbID : lhcbid_field {};
struct Hits : vector_field<struct_field<Index, LHCbID>> {};

// Define collection:
struct Tracks :
     SOACollection <Tracks, Momentum, Hits>{};
```

Struct of Arrays vs Array of Structs (traditional model)



*Mniszewki et al, Int.J.HighPerformanceComputing Applications **35** (2021)*

Suports SIMD or scalar backends using Proxies

- one object is a slice in the collection
- a proxy represents a chunk of N objects in the collection
- in case of scalar N is 1
- in case of SIMD N is width of SIMD vectors.

Vector

```
// Push N elements to the end of tracks
auto proxy = tracks.emplace_back <simd >();
proxy.field<Momentum>().set(momentum);
// Iterate over tracks N elements at a time
for (const auto& proxy : tracks.simd()) {
auto momentum = proxy.get<Momentum>();
```

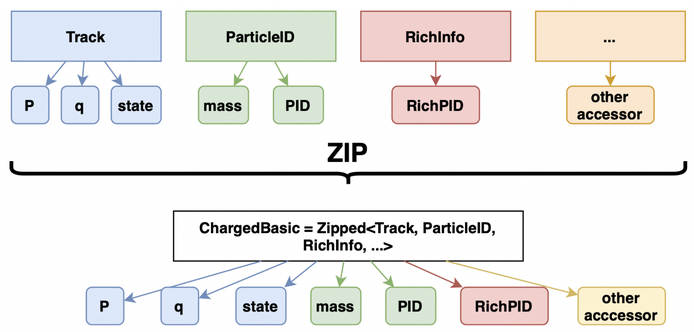
Scalar

```
// Push 1 element to the end of tracks, possibly masking it
// Set the momentum of the track
auto proxy = tracks.emplace_back <scalar>(mask);
proxy.field<Momentum>().set(momentum);

// Iterate over tracks one at a time
for (const auto& proxy : tracks.scalar())
    auto momentum = proxy.get<Momentum>();
```

Connecting SOACollections

Zipping - Same size SOA



N.Nolte CERN-THESIS-2020-331

SOARelations

- Connecting elements in a collection to other information.
- SOARelations are SOACollections representing relations between SOACollections

```
struct TracksPVsRelWithWeight:
     RelationTable2D<Tracks, PVs, Weight>{};
TracksPVsRelWithWeight table {tracks,  pvs};
auto proxy = table.emplace_back<simd>();
proxy.set(tracks.indices(), pvs.indices(), weight);
```

SIMD Wrappers

- Intrinsic for vector operations depending on architecture
- LHCb specific SIMDWrapper developed

```
// scalar
scalar::float_v min( scalar::float_v lhs, scalar::float_v rhs ) {
    return std::min( lhs.data, rhs.data );
}
// neon
neon::float_v min( neon::float_v lhs, neon::float_v rhs ) {
    return vminq_f32( lhs, rhs );
}
// avx
avx::float_v min( avx::float_v lhs, avx::float_v rhs ) {
    return _mm256_min_ps( lhs, rhs );
}
```

Throughput Oriented (ThOr) selections

- Functors designed to be agnostic about Input and Output
- Use of functor cache instead of JIT compilation
- Significant gain using SIMD instructions

Implementation	$D^+ \to K^+\pi^+\pi^-$ execution time
CombineParticles	256 µs
NBodyDecays	77.1 µs
ThorParticleCombiner	38.8 µs
ThOrCombiner Scalar	10.2 µs
ThOrCombiner SSE	7.5 µs
ThOrCombiner AVX2	6.9 µs

CERN-THESIS-2020-331

Persistency

- Filter and select what needs to be persisted
- Create persistent representations
- SOACollections are already in persistent representations

```
// Define tags :
struct Momentum : float_field {
using packer_t = SOAPackFloatAs <short,
                                 std::ratio<1, 100 > >;
};
struct Unwanted : int_field {
   using packer_t = SOADontPack ;
};

struct OopsIForgotThisField : int_field {
using packer_t =
    SOAPackIfVersionNewerOrEqual<1, SOAPackNumeric <int>>;
};
// Define collection :
struct Tracks : SOACollection<Tracks,Momentum,
                              Unwanted, OopsIForgotThisField> {};
```

Summary

Example of the Event Model applied to the HLT1

Evolution of the current LHCb HLT1 throughput of a CPU prototype from Dec-2018 to Apr 2020

LHCb-FIGURE-2020-007