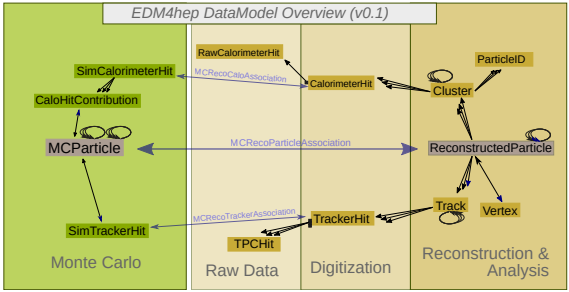
podio - latest developments and new features of a flexible EDM toolkit

HSF WLCG Virtual Workshop

Thomas Madlener
on behalf of the Key4Hep Software Group

Nov 24, 2020

EDM - Event Data Model



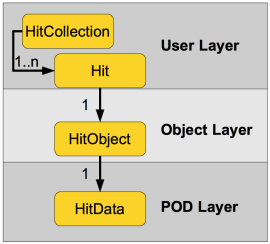
key4hep/EDM4Hep

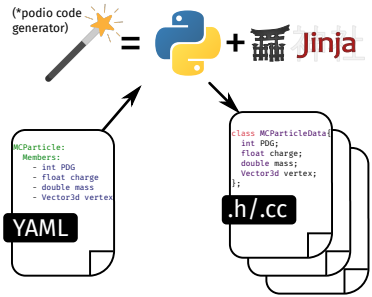
EDM4Hep is using podio
Both are part of the Key4Hep SW stack

- The EDM defines the vocabulary different components use to talk to each other
- Defining an EDM is not entirely trivial
- Implementing one is not easy either
  - Relations between objects
  - Efficient implementation and I/O
  - Thread safe
  - Easy to use™

podio - Basic Design

- **User Layer** consists of handles to the EDM objects and offers the full functionality
  - Offers value semantics making it easy to use
- The **Object Layer** handles resources and references to other EDM objects
- The actual POD (plain old data) structs live in the **POD Layer**
- Layered design allows for efficient memory layout and performant I/O implementation
- The **podio code generator** generates all the necessary c++ code from a high level description
- CMake macros facilitate downstream usage



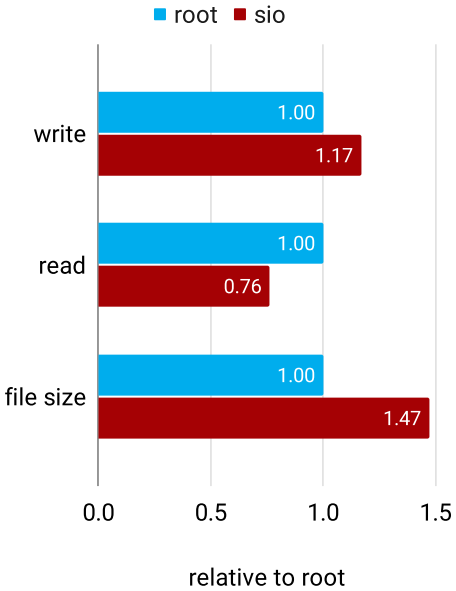


AIDASoft/podio

SIO Backend

NEW

- Added possibility to use `sio` library for persistency
  - Used for persistency in `LCIO`, the EDM in `iLCSoft`
  - iLCSoft/SIO
- Can be used as a drop-in replacement for the default `root` based I/O layer
- **Preliminary benchmarks** using `EDM4Hep` and $ee \to ZH \to \mu\mu X$ (Higgs recoil) @ILD
  - `root` faster for writing and with smaller file sizes
  - `sio` faster for reading and with less variation in per-event times
- Confirm that podio design is able to support multiple backends



Metadata

Definition of *metadata*

: data that provides information about other data

NEW

- Necessary data to correctly interpret information from the EDM
  - (Non-constant) encoding information
  - No clear border of what metadata actually comprises
- Capabilities deliberately limited
  - Avoid having an “EDM inside your EDM” built on top of metadata
  - Fix your EDM - do not work around it
- **Prototype** implementation is present and useable
  - Thread safety not yet completely solved
- Available at the **Run**, **Collection** and **Event** level

```
using namespace std;
template<typename T>
using MapType = map<string, vector<T>>;

class GenericParameters {
public:
  // some get/set functionality

private:
  MapType<int> _intMap;
  MapType<float> _floatMap;
  MapType<string> _stringMap;
};

// usage:
auto& md = store.getEventMetaData();
md.setFloatVal("weight", 1.234);
// reading it
auto weight = md.getFloatVal("weight");
```

Next Steps and Goals

- Make podio more self-contained
  - Provide higher level functionality outside frameworks like Gaudi
- Tackle thread-safety and thread-policy for new features
- Increase set of benchmarks
- Follow up on open github issues and feature requests
  - Feel free to add ideas and requests on AIDASoft/podio

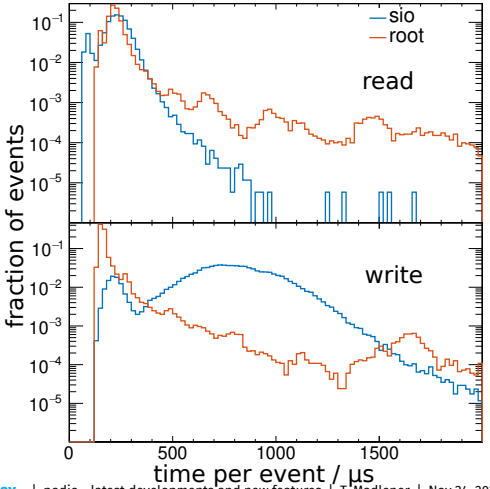
Thank you for your attention!

Questions?

Backup

Preliminary benchmark details

- Using `delphes` and `EDM4Hep` on physics scenario: $ee \to ZH \to \mu\mu X$ (Higgs recoil) @ILD
  - $\sim$ 150 `MCParticle`, $\sim$ 50 `ReconstructedParticle`, $\sim$ 50 MC - Reco associations per event
  - Comparable to (mini-)**DST** or (Mini)**AOD** data tiers; approx. 17k events



	root	sio
read		
total [s]	5.1	3.9
setup & teardown [ms]	182	2.8
per event [µs]		
median	220	230
90 %	285	300
max	4.6e5	2.0e3

write		
total [s]	11.4	13.3
setup & teardown [ms]	1490	23
per event [µs]		
median	160	770
90 %	280	920
max	2.5e6	1.1e4

CMake Macros

NEW

(Almost verbatim from EDM4Hep)

```
find_package(podio)

# Generate the c++ source files for all I/O backends that the installed podio supports
PODIO_GENERATE_DATAMODEL(edm4hep ../edm4hep.yaml headers sources IO_BACKEND_HANDLERS
↪  ${PODIO_IO_HANDLERS})

# Define the core datamodel lib as target
PODIO_ADD_DATAMODEL_CORE_LIB(edm4hep "${headers}" "${sources}")

# Define the root I/O dictionaries as target (*conditionally)
PODIO_ADD_ROOT_IO_DICT(edm4hepDict edm4hep "${headers}" src/selection.xml)

# Define the sio I/O handling blocks as target (*conditionally)
PODIO_ADD_SIO_IO_BLOCKS(edm4hep "${headers}" "${sources}")

# Can check if a target exists and then use it
IF(TARGET edm4hepSioBlocks)
  message(STATUS "SIO backend is supported")
ENDIF()
```

`(*conditionally)`: These macros will always work. However, targets will only be defined if `podio` supports the backend