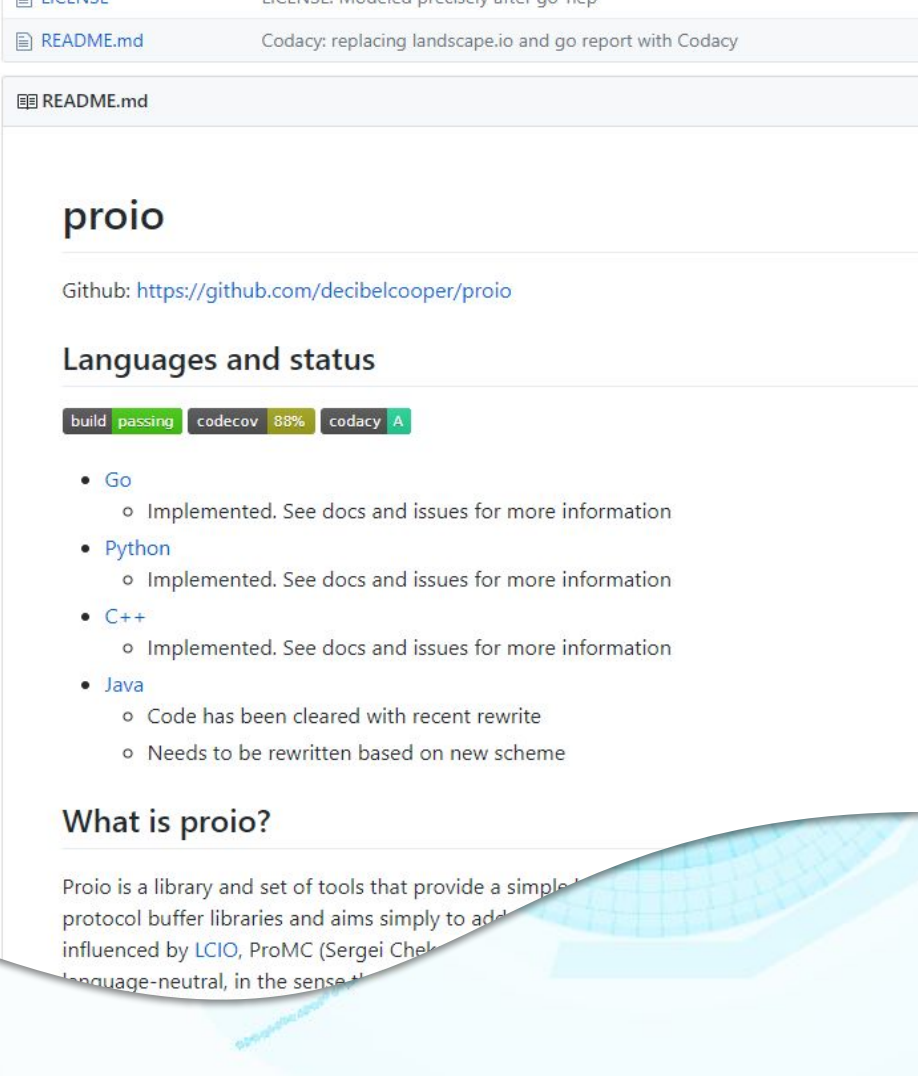


# ProIO

David Blyth

# The Project

- Inspired by works from S. Chekanov and A. Kiselev
- Lives at <https://github.com/decibelcooper/proio>
- Ooh, shiny badges!
  - Continuous Integration: no code merges without sufficient testing.
  - Unit test coverage goal is to maintain > 90%
  - Automated code “quality” checks
- Contributions of *all* kinds are welcome.



LICENSE: modified precisely after go help

README.md Codacy: replacing landscape.io and go report with Codacy

README.md

## proio

Github: <https://github.com/decibelcooper/proio>

### Languages and status

build passing codecov 88% codacy A

- [Go](#)
  - Implemented. See docs and issues for more information
- [Python](#)
  - Implemented. See docs and issues for more information
- [C++](#)
  - Implemented. See docs and issues for more information
- [Java](#)
  - Code has been cleared with recent rewrite
  - Needs to be rewritten based on new scheme

### What is proio?

Proio is a library and set of tools that provide a simple protocol buffer libraries and aims simply to add... influenced by [LCIO](#), ProMC (Sergei Chekanov) and... language-neutral, in the sense that...

# ProIO Key Concepts

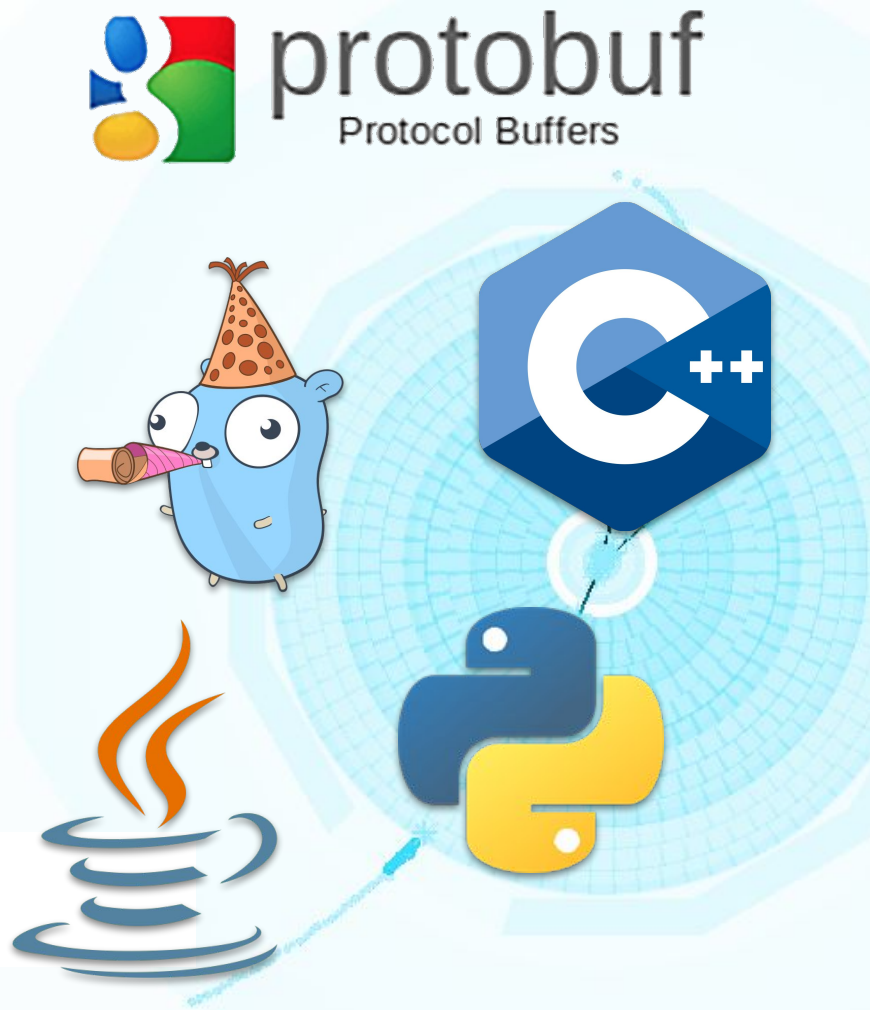
- ProIO is for PROS! It's right in the name...

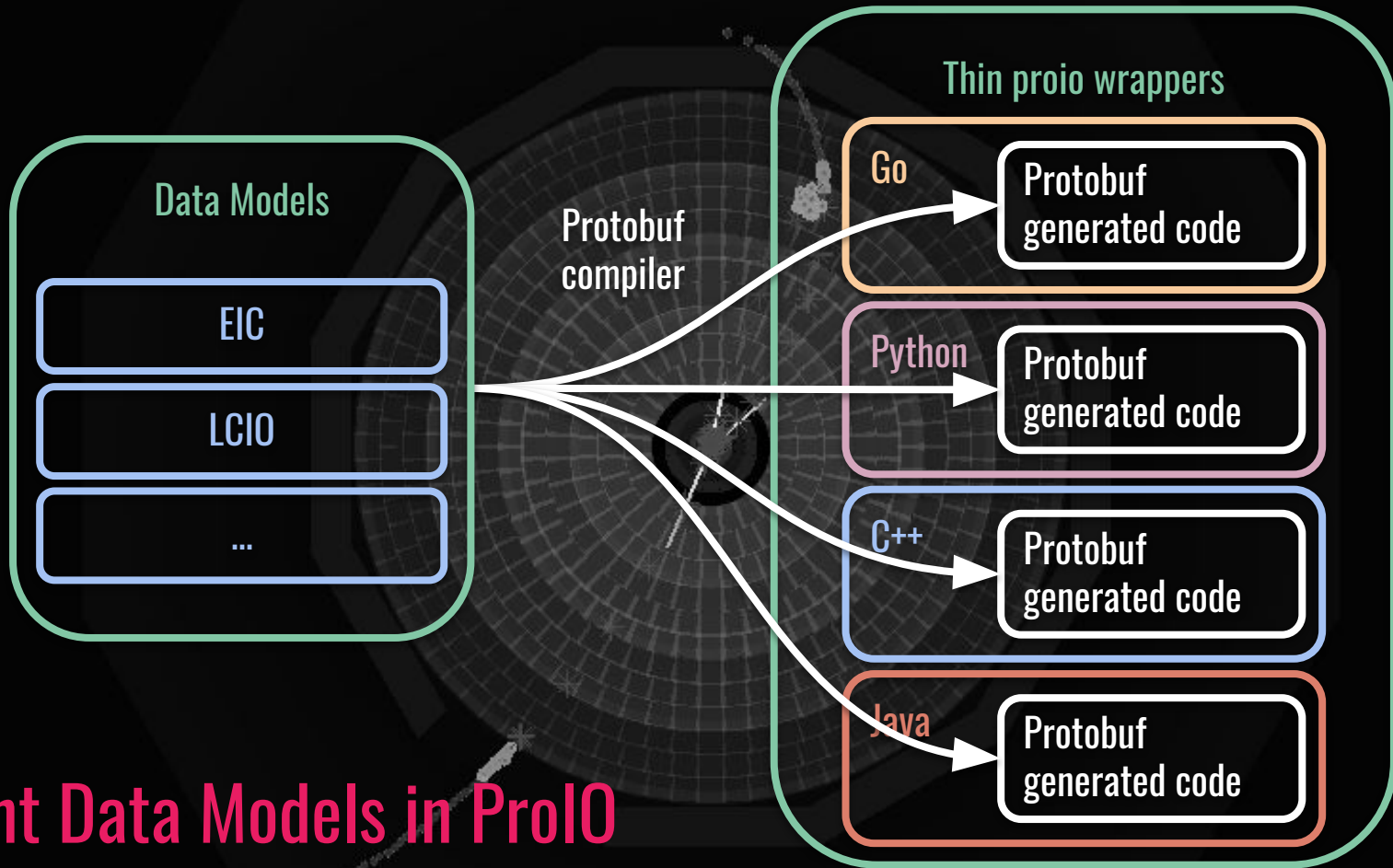
...J.K., the name has nothing to do with that, and everything to do with Google's Protocol Buffers (Protobuf)



# ProIO Key Concepts

- Language-neutral I/O for streaming events
- Thin, native containers for protobuf messages, simply adding the concept of an event
- `protobuf + event structure = ProIO`
- Serialized output can be accessed effectively in archival file, or in a stream





## Event Data Models in ProIO

# Data Model Messages

- Pure [protobuf messages](#)
- Written in a syntax that is simple and familiar
- Can be modified and added to without writing any language-specific code
- Does NOT have to be part of ProIO repo!

```
Branch: master | proio / model / eic.proto
```

decibelcooper EIC data model: added Track object for testing

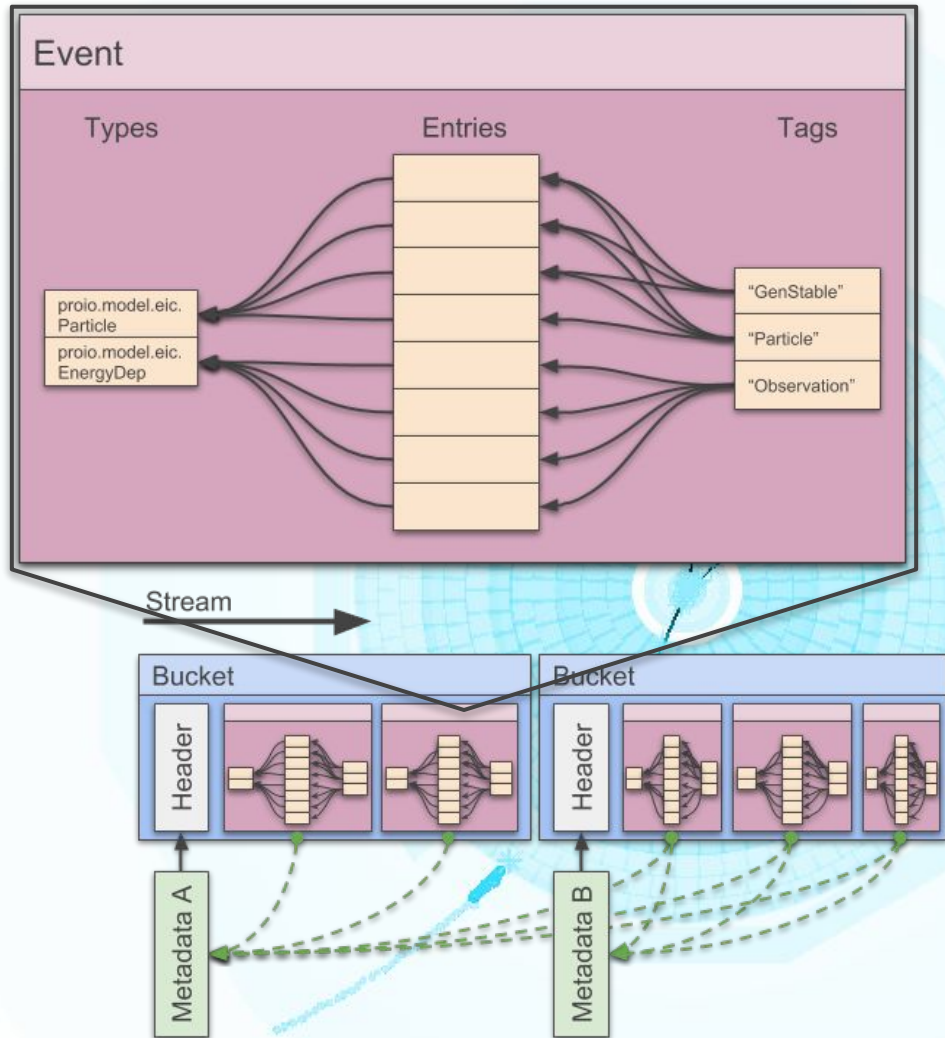
1 contributor

131 lines (114 sloc) | 3.41 KB

```
1  syntax = "proto3";
2  package proio.model.eic;
3  option go_package = "github.com/decibelcooper/proio/go-proio/model/eic";
4  option java_package = "proio.model";
5  option java_outer_classname = "Eic";
6
7  ///// TRUTH LEVEL DATA MODEL MESSAGES /////
8
9  message Particle {
10     // ProIO entry identifiers that point to parent Particles
11     repeated uint64 parent = 1;
12     // ProIO entry identifiers that point to child Particles
13     repeated uint64 child = 2;
14     // PDG code
15     sint32 pdg = 3;
16     // position in mm and time in ns
17     XYZTD vertex = 4;
18     // momentum in GeV
19     XYZD p = 5;
20     // mass in GeV
21     double mass = 6;
22     // charge in units of e
23     float charge = 7;
24     XYZF spin = 8;
25 }
26
27 ///// SIMULATION LEVEL DATA MODEL MESSAGES /////
```

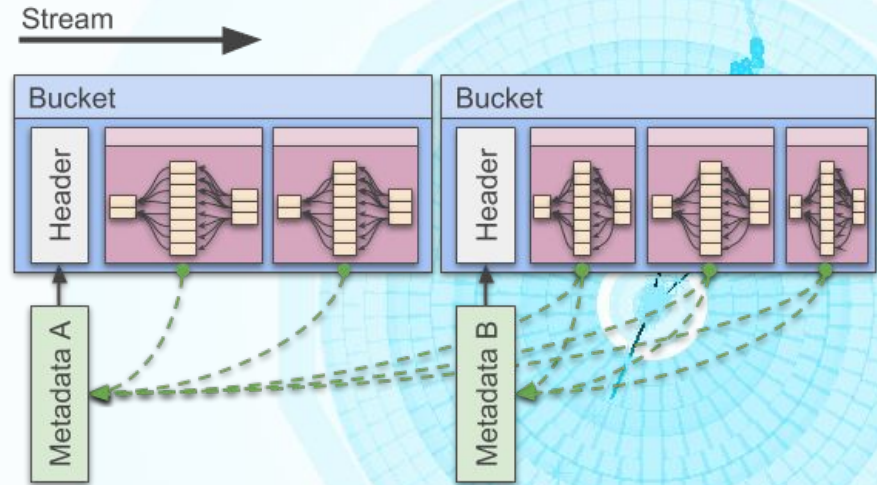
# Event Structure

- **Entries**
  - Each entry is an arbitrary protobuf message with a unique, persistent ID
- **Tags**
  - Primary means of (non-linear) event data organization
  - Each tag is a mapping from a string to a list of entry IDs
- **Metadata**
  - Key-value pairs that are shared among events



# Bucket Structure

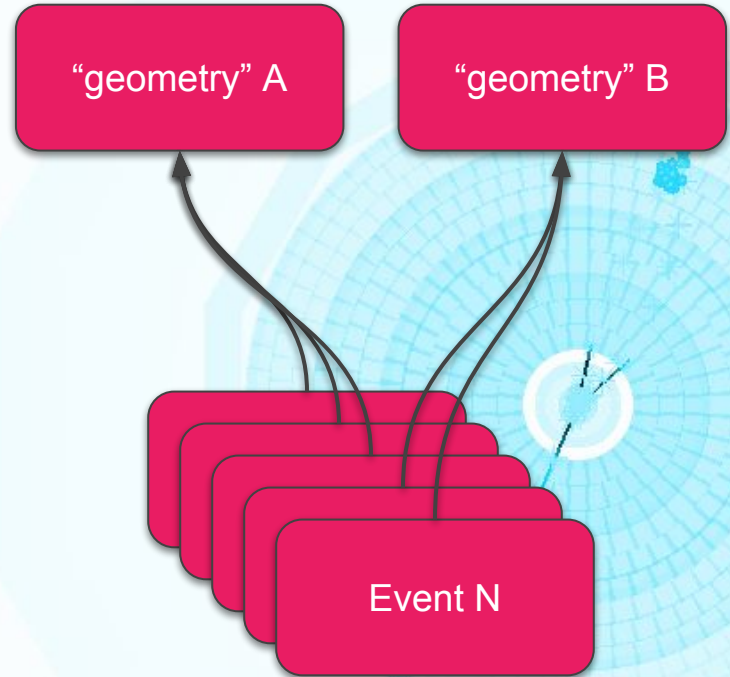
- Buckets are the quantum of ProIO data “on the wire”
- Configurable for payload size and **compression** type (gzip, lz4, or none)
- Carries **metadata** to be attached to events
  - Metadata stored as key-value pairs
  - Each key-value pair is associated with all future events until it is overridden
- Provides **resynchronization** in the case of corrupt data





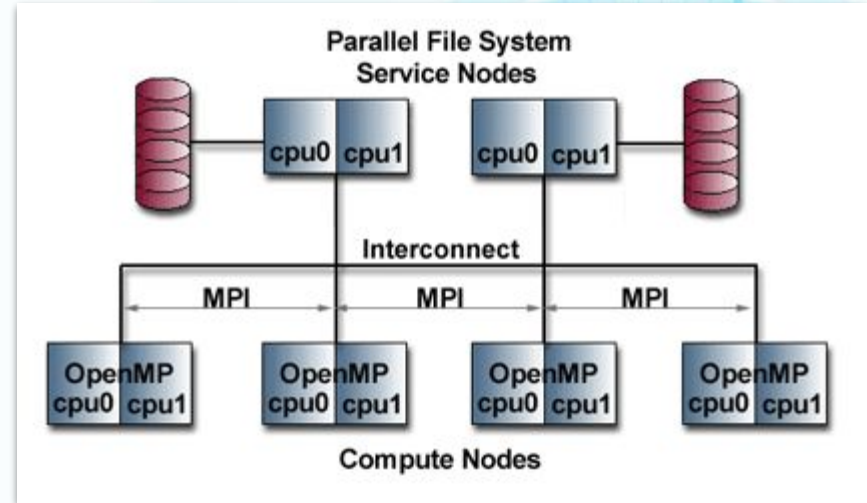
# Metadata

- Intended to support things like attaching MC parameters, GDML, and magnetic field configurations
- Like with event entry tagging, adoption of conventions for EIC is encouraged.
- E.g., GDML may be injected into the ProIO stream with the “geometry” key.
- Reconstruction should watch for this key to be attached to events.



# Notes on MPI

- Any HPC administrator will push the use of message passing.
  - They have good reasons for this.
- MPI can benefit from an event container that is **self-serializing**.
- Protobuf and ProIO provide, IMO, an elegant solution to this
  - ProIO events have value even if we don't use ProIO streams.



# Command Line Tools

- Written in Go
  - `proio-summary`
  - `proio-ls`
  - `proio-strip`
  - `lcio2proio`

Try these out by pulling

`docker://electronioncollider/ani-base,`

or by setting up a simple Go environment and doing a “go get”:

```
go get github.com/decibelcooper/proio/go-proio/...
```

# Future Work

- Last bits of APIs are being added in near future, but are nearly stable right now.
  - Note: ProIO data are already stable! Last bits of API functionality will not break this!
- Proposed JLab LDRD may put ProIO to the test in a streaming readout context
- Summer student will work on
  - Graphical data browser implemented in Python
  - Generating MC events directly into ProIO

