Event Data Model Project

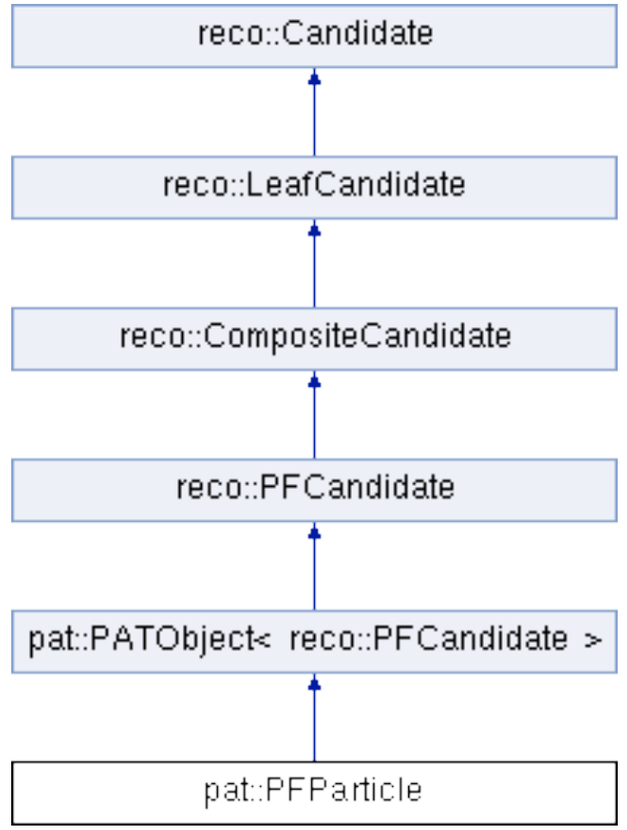
WP 3.4

Benedikt Hegner
CERN

AIDA-2020 Kick-Off Meeting
3.6.2015

Data Models

- LHC experiments show that we overdid on inheritance and polymorphism
  - State-of-the-art when code was written
- Sometimes **deep and absurd object hierarchies**
  - A "CaloTower" as "Candidate" in CMS
- Most physicists do not feel productive in the existing data models
  - Many n-tuple based 'frameworks'
- During the last 10 years technology evolved a lot!
- ***There is a need to rethink what we did...***



Data Model Implementations

- **Almost every experiment/project writes its own data model implementation**
  - Waste of resources
  - Little interoperability across projects
- **LCIO tried to address this issue with a common data model for linear collider studies**
  - Turned out to be a key element to the success of the LC software
- **Different physics use cases require different data models though**
  - Convergence on a single model like LCIO would be very hard once going beyond ee-physics
- **Can one still share models and implementations... ?**

Data Model Library

- **Aim is a toolkit to define and create efficient data models**
  - Tailor it to your use case when needed
  - Share the data definition across projects wherever possible
  - Benefit from other people's expertise (and debugging)
- **Interest already expressed by**
  - LC - for the next evolutionary step of LCIO
  - FCC - for the data model of FCC-ee, -eh, and -hh
- **Could serve as common denominator for other projects**
  - The data model of a reconstruction library could become natural part of an experiments overall data model

Technical Work

The PODIO prototype

Technical Considerations

- **Simple memory model**
  - Employ **simple structs (PODs)** instead of fat objects
  - Allow for data access for vectorization
- **Simple class hierarchies**
  - Wherever possible use **concrete types**
  - Favor composition over inheritance
- **Simple I/O setup**
  - Keep transient to persistent layer as thin as possible
  - Whenever possible store PODs as is
- **Support for multi-threading**
  - should integrate well w/ multi-threaded frameworks
- **Simple model generation**
  - Employ code generation to make it easy for the user
  - Quick turn-around for improvement on the back-end

Interlude: what's a POD?

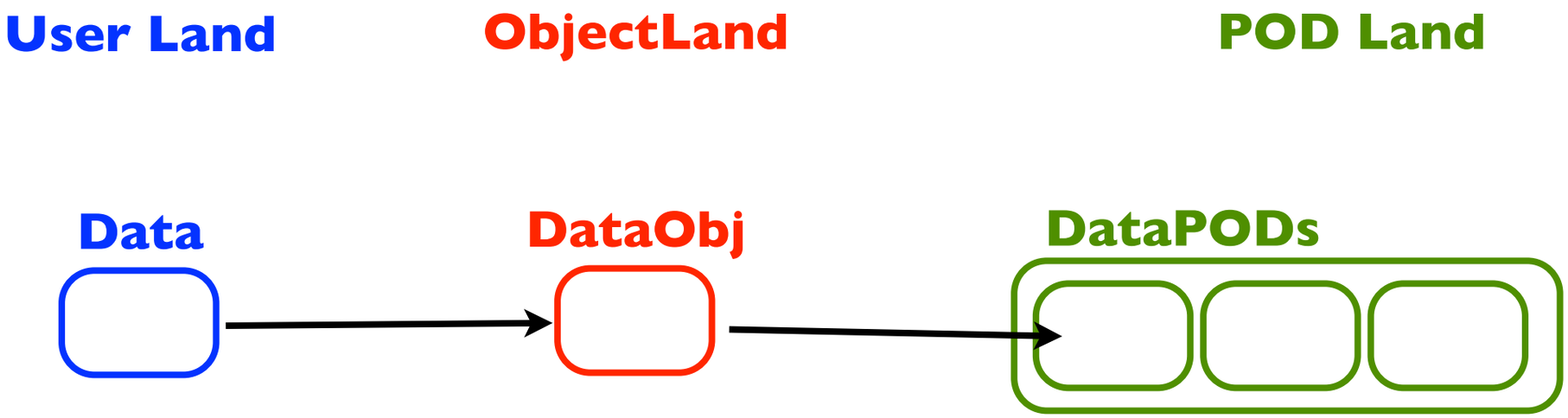
- **In C++11 A POD combines two concepts**
  - Support for static inititalization (i.e. a *trivial class*)
    - E.g. No custom constructors, and destructor
  - They have *standard-layout*
    - no virtual functions and no virtual base classes,
    - same access control (public,private,protected) for all non-static data members,
    - ...
  - A struct is the most prominent example of a POD

- **A POD is good for memory layout, and memory operations**

Separation of Concerns

- **Using PODs is a good idea, but...**
  ... they are a little bit too dumb to support all what is needed.

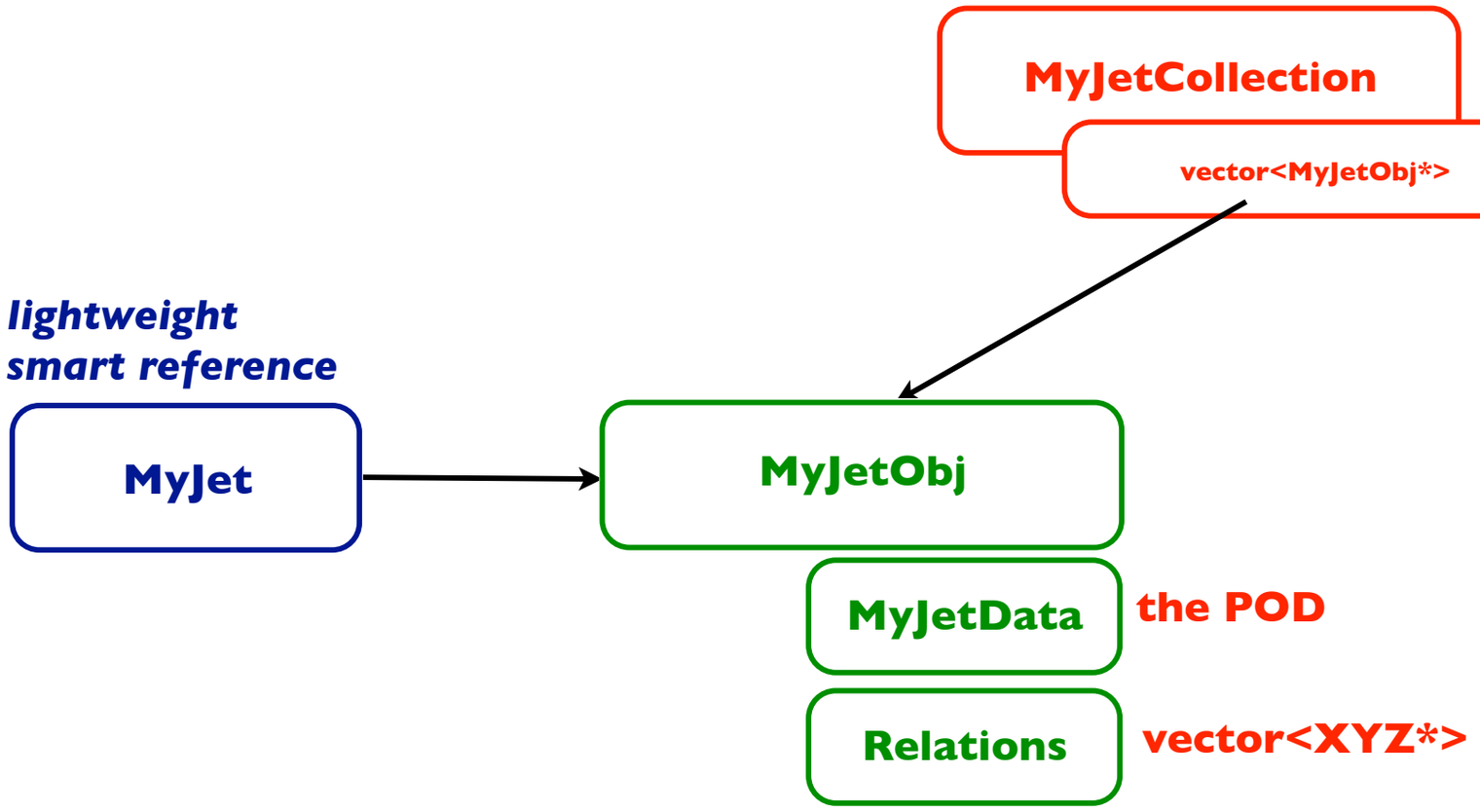
- **Need smart layers on top of the PODs**
  - Dealing with ownership
  - Allow referencing between objects
  - Deal with non-trivial I/O operations



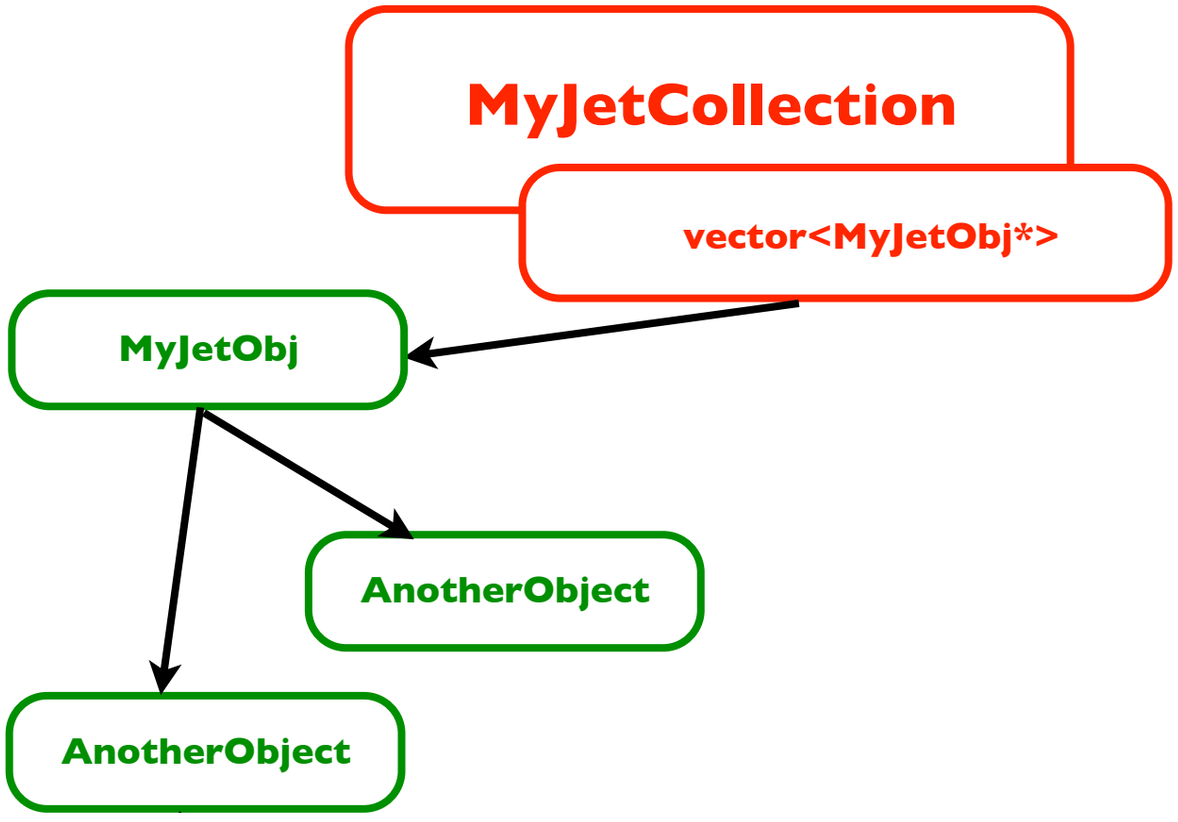
- **Whenever really performance critical - leave possibility of access to bare PODs**

References between objects

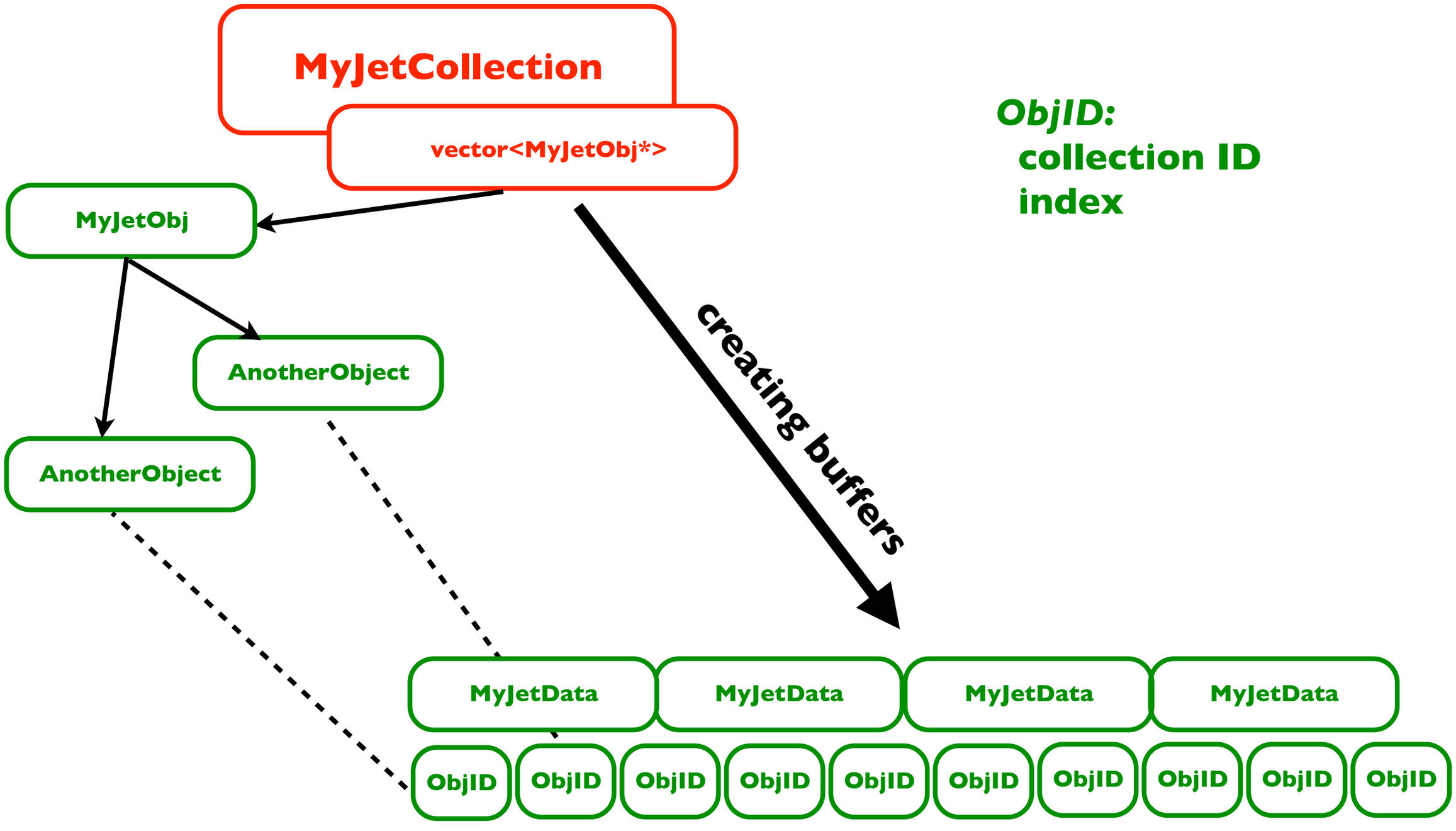
- **There is always a need of having references between objects**
  - E.g. a jet knowing about its components
- **The "Object Land" manages the lookup in memory**
- **Relations are handled outside the PODs**



POD and I/O - PODIO



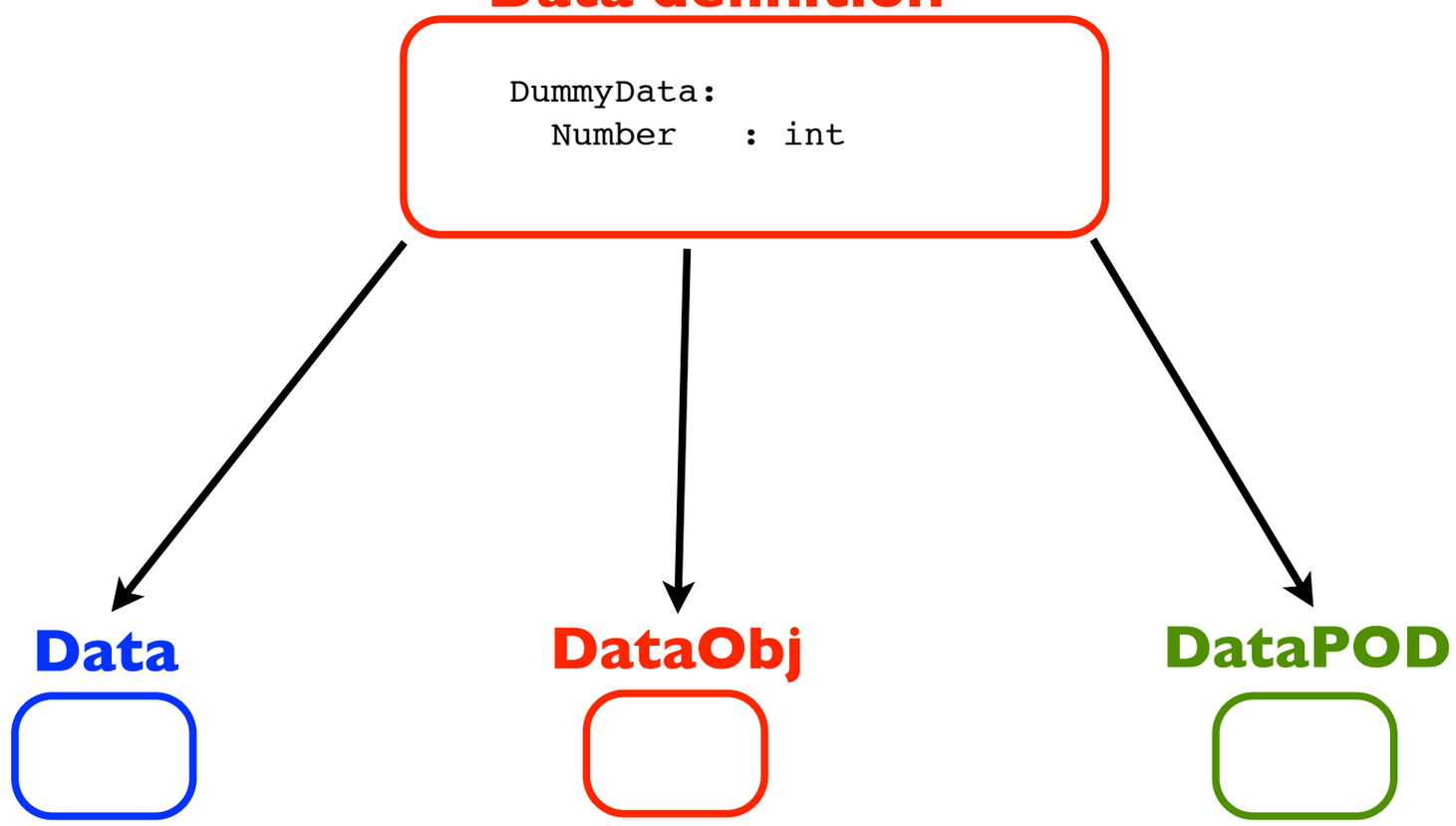
POD and I/O - PODIO



ObjID:
collection ID
index

Code generation

Automatic creation of the 3 data views



Data Model Definition

simple members:

```
RawCalorimeterHit:
  description : "raw calorimeter hit"
  author : "B. Hegner"
  members :
    - int cellID0 // The detector specific (geometrical) cell id.
    - int amplitude // The amplitude of the hit in ADC counts.
    - int timeStamp // The time stamp for the hit.
```

relation to one other object:

```
SimCalorimeterHit:
  description : "sim calorimeter hit"
  ...
  OneToOneRelations :
    - MCParticle particle // The MCParticle that caused the hit.
```

relation to many other objects:

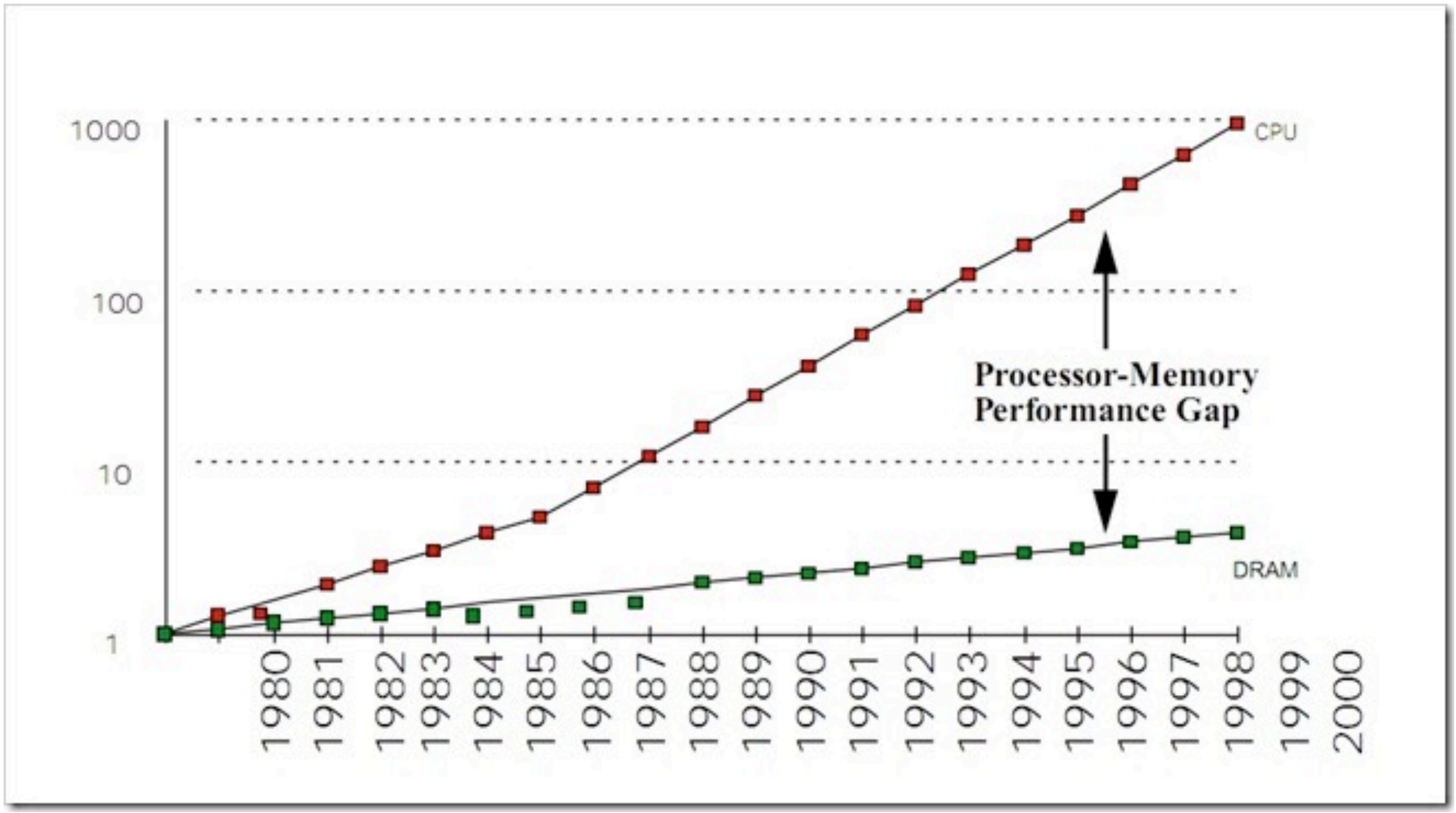
```
MCParticle:
   ...
  OneToManyRelations :
    - MCParticle daughters // The daughters of this particle
    - MCParticle parents   // The parents of this particle
```

Summary

- **Aim is a toolkit to define and create efficient data models**
- **Going for a simplistic model w/ plain-old-data and simple data types**
- **Current prospective clients**
  - LC - for the next evolutionary step of LCIO
  - FCC - for the data model of FCC-ee, -eh, and -hh
- **Prototyping work just started**
  - https://github.com/hegner/podio
  - Outcome will be a little design document
- **Current contributors**
  - Frank Gaede (DESY)
  - Pere Mato (CERN)
  - Benedikt Hegner (CERN)

Backup

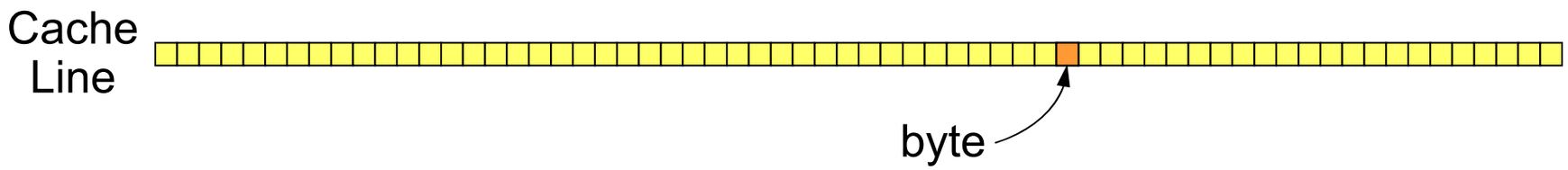
Memory Speed Development



More than a factor 100 !

Caching

- Caching is - at distance - no black magic
- Usually just holds content of recently accessed memory locations



- Caching hierarchies are rather common:
  - 32KB L1 I-cache, 32KB L1 D-cache per core
    - ➡ Shared by 2 HW threads
  - 256 KB L2 cache per core
    - ➡ Holds both instructions and data
    - ➡ Shared by 2 HW threads
  - 8MB L3 cache
    - ➡ Holds both instructions and data
    - ➡ Shared by 4 cores (8 HW threads)

Very tiny compared to main memory!

Addressing Thread-Safety

- **Whatever new data model library needs to think of multi-threaded environment**
  - Multiple access to same data
  - Process ***multiple events concurrently***
- **Problems w/ multi-threading**
  - Non-const objects
  - Internal caching
  - On-demand reading from disk
- **Possibilities w/ multi-threading**
  - Pipeline persistent-to-transient and transient-to-persistent operations
- **All this is part of the prototype activity**