# Status and strategy for the FCC Software

Colin Bernet (IPN Lyon)
Benedikt Hegner (CERN)

FCC-ee workshop, Oct 28 2014

# Early SW decisions

- Pragmatism counts
- Adapt existing solutions from LHC:
  - Gaudi as underlying framework
  - ROOT for I/O
  - Geant4 for simulation
  - **Python for user analysis (heppy)**
- Adapt software developments from ILC/CLIC
  - DD4Hep for detector description
- Invest in better fast vs. full sim integration
  - Geant4 fastsim, Atlfast
  - More later in this session
- **Invest in proper data model**
  - The LHC experiments' ones are over-engineered
  - The ILC/CLIC implementation isn't state of the art
  - Both are significantly under-performant on modern CPUs

# Event data model

# The Design Principle is Simplicity

- Easy for the user 
  - No deep hierarchies
  - Hide the details
  - Code generation from simple description
- Easy for the CPU 
  - Avoid virtual calls
  - Usable in hybrid C++/ Python setup
  - Very fast
  - Future-proof

This brings us from smart objects back to PODs…

# POD : Plain Old Data

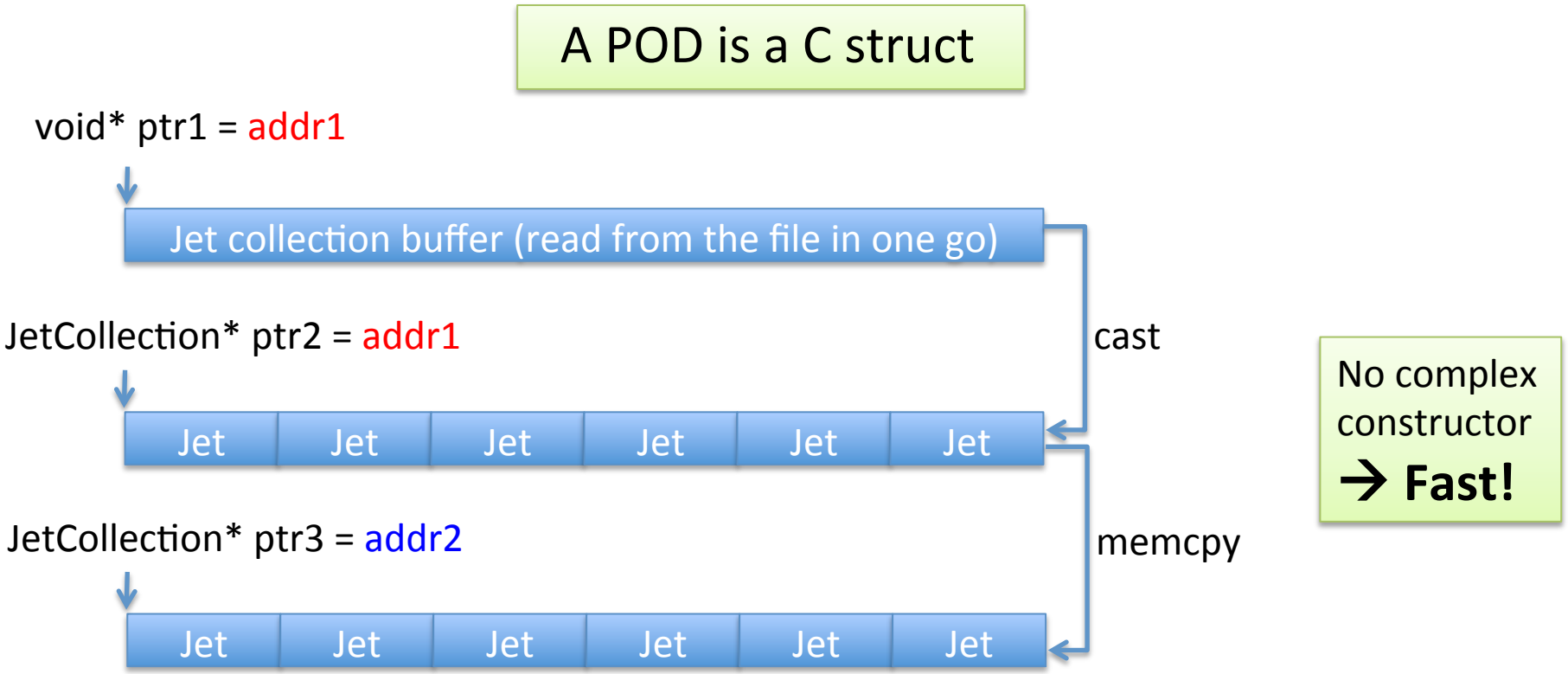
**Definition**

A POD struct is a non-union class that is both a trivial class and a standard-layout class, and has no non-static data members of type non-POD struct, non-POD union (or array of such types). Similarly, a POD union is a union that is both a trivial class and a standard layout class, and has no non-static data members of type non-POD struct, non-POD union (or array of such types). A POD class is a class that is either a POD struct or a POD union.

just kidding ;-)

# POD : Plain Old Data

**Definition (slightly easier and more or less correct)**



# Other benefits of PODs

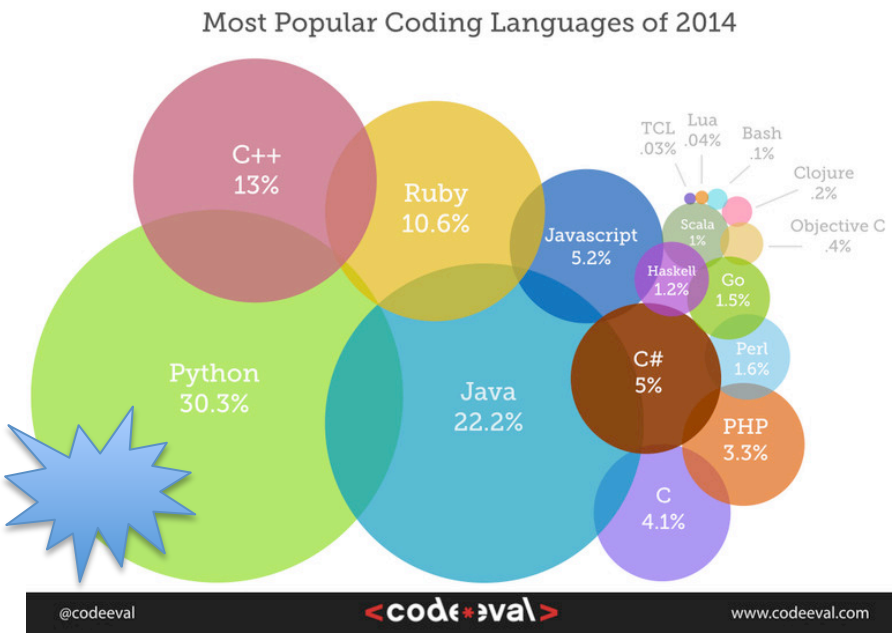
- I/O is a trivial operation
  - Does not rely on fancy high-level features of ROOT, etc
- Working with simple types avoids costly virtual calls
  - On adding two four vectors time is not wasted in the real **add**
  - Time is wasted in accessing data
- PODs can eventually be used on accelerators (GPUs)
- However they don't support all what is needed
  - Object navigation/cross-linking as one example
- We augment the PODs by a small layer in user land
  - You don't see the POD if you don't want to
  - But the compiler does and can optimize properly
  - Paves way for parallelization (vectorization)
- More details in presentations given in the weekly software meeting

# The open question of the data model

- Implementation road clear and large fraction done
- But what should the defined data types contain in terms of members/physics?
  - Detector design independent
  - Simple and versatile
- ILC successfully tackled that problem with LCIO
  - We will probably start with their choice contained data
- ILC very interested in our implementation and watching closely
  - First example for bi-directional benefit ?!

# Heppy

# Why python?




- Very large user base
- Super easy to learn
- Light & short code
- Good performance
  - usually wraps C or C++ modules
- « Batteries included »
  - massive and easy-to-use standard library
- Dynamic typing
  - good for multichannel analyses
  - code highly reusable
- Dynamic object modification
  - Can attach new attributes (or methods) to an existing object
- Productivity x 5-10 w/r C++
- A lot of fun!

```
import PhysicsTools.HeppyCore.framework.config as cfg
from PhysicsTools.HeppyCore.framework.chain import Chain as Events
from PhysicsTools.HeppyCore.analyzers.Printer import Printer
from PhysicsTools.HeppyCore.analyzers.SimpleTreeProducer import SimpleTreeProducer

# input component
# several input components can be declared,
# and added to the list of selected components
inputSample = cfg.Component(
    'test_component',
    files = ['test_tree.root'],
    # tree_name = 'test_tree'
    )

selectedComponents  = [inputSample]

printer = cfg.Analyzer(
    Printer
    )

tree = cfg.Analyzer(
    SimpleTreeProducer,
    tree_name = 'tree',
    tree_title = 'A test tree'
    )

# definition of a sequence of analyzers,
# the analyzers will process each event in this order
sequence = cfg.Sequence( [
    printer,
    tree
    ] )

# finalization of the configuration object.
config = cfg.Config( components = selectedComponents,
                     sequence = sequence,
                     events_class = Events )
```

Run with:
**heppy.py OutDir/ my_cfg.py –N 100**
Can run and manage 100s of jobs in a couple commands

Definition of an input component (sample)
Passed to all analyzers

Analyzer configuration fragments.
Passed to the corresponding analyzer

Scheduling sequence

Process configuration

# Dummy JetAnalyzer

```
from PhysicsTools.HeppyCore.framework.analyzer import Analyzer
from PhysicsTools.HeppyCore.utils.deltar import matchObjectCollection
from JetClasses import Jet, GenJet


class JetAnalyzer(Analyzer):

    def process(self, event):
        jets = map( Jet, event.input.jets )
        event.jets = [ jet for jet in jets if jet.pt() > 30 ]
        event.genjets = map( GenJet, event.input.genjets )
        matches = matchObjectCollection(event.jets,
                                        event.genjets)
        for jet in event.jets:
            jet.genJet = matches[jet]
```

# Status and Plans

- Example chain working:
  - Reading HepMC
  - Jet clustering inside Gaudi
  - Writing out a data file
  - Reading the data file with Python (heppy)
  - Plotting

- Now this is waiting for **your ideas** to be added
  - The tutorial today is your first step towards it

# Organization and Communication

# Going full steam at quite a few places…

- Weekly SW meetings with significant attendance:
  - https://indico.cern.ch/category/5666/
- Lots of things ongoing in data model, python analysis and simulation
- A good team of highly motivated people!
- If there is a bug or a rough edge it may be gone before you even realize ☺

# …and not so at a few others…

- Framework
  - Core event data model, Gaudi integration, Software stack

Bernet, Hegner

- Generators
  - Integration

**People needed**

- Simulation infrastructure
  - Geant-4 (fast & full)

Carminati, Dell'aqua, Hrdinka, Salzburger, Williams, Zaborowska (Convener: Ribon)

  - Delphes integration

Hegner, **People needed**

- Reconstruction

**People needed**

- Analysis tools
  - python & C++ framework

Bernet

- Validation
  - testing and performance

Lukas Marti

- Computing
  - sample production and management

**People needed**