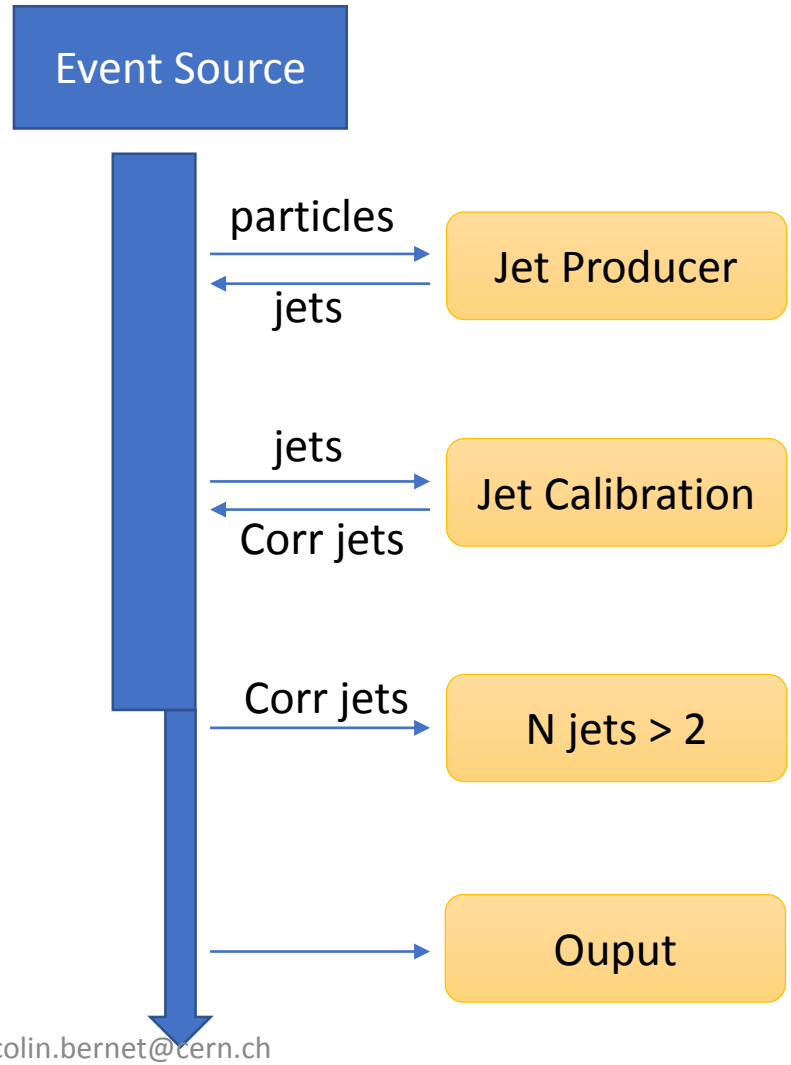
# Heppy

A Lighweight Event Processing Framework
for High Energy Physics in python

Colin Bernet (IPNL)

colin.bernet@cern.ch

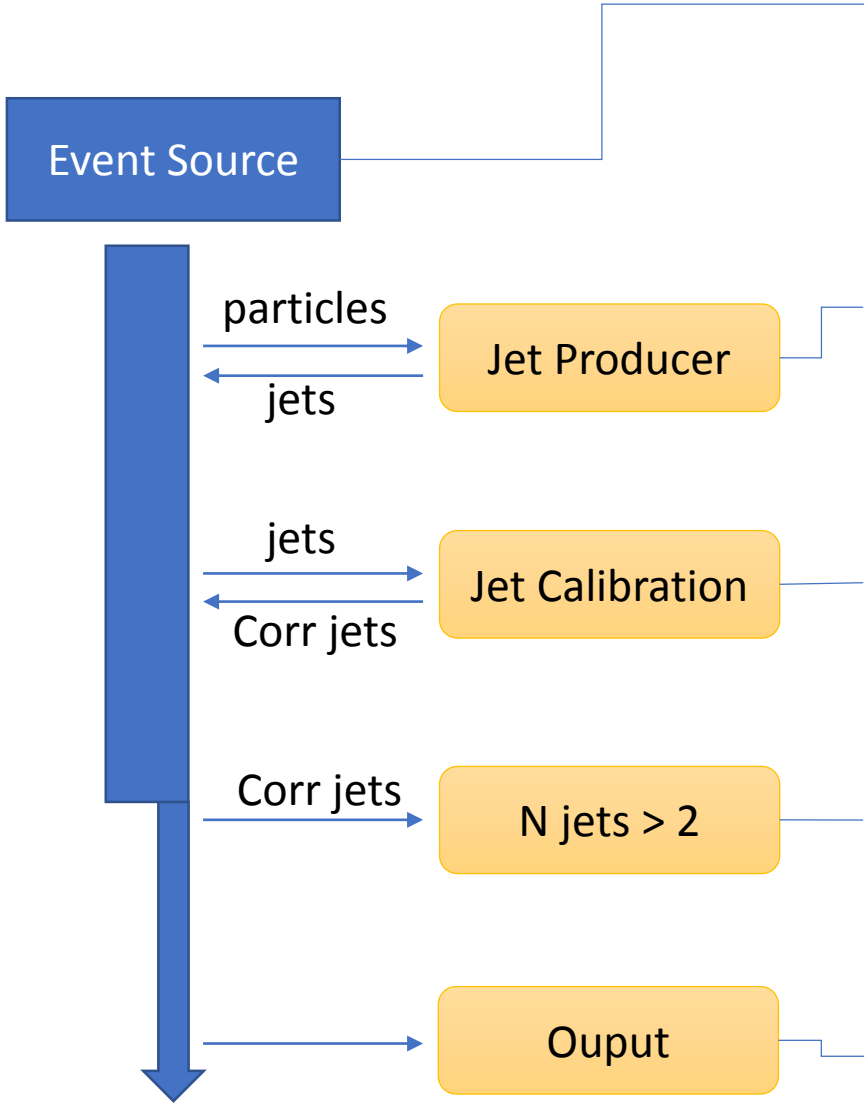
# Event Processing Frameworks




- e.g.
  - Gaudi (ATLAS, LHCb, FCC)
  - CMSSW (CMS)
  - Marlin (ILC/CLIC)

- All written in C++
- Too heavy for analysis

# Heppy




**Input can be anything:**
- Event or collection (e.g. jets)
- Any format (backends):
  - CMS, ILC, plain root, text, …
- No input (generator mode) possible

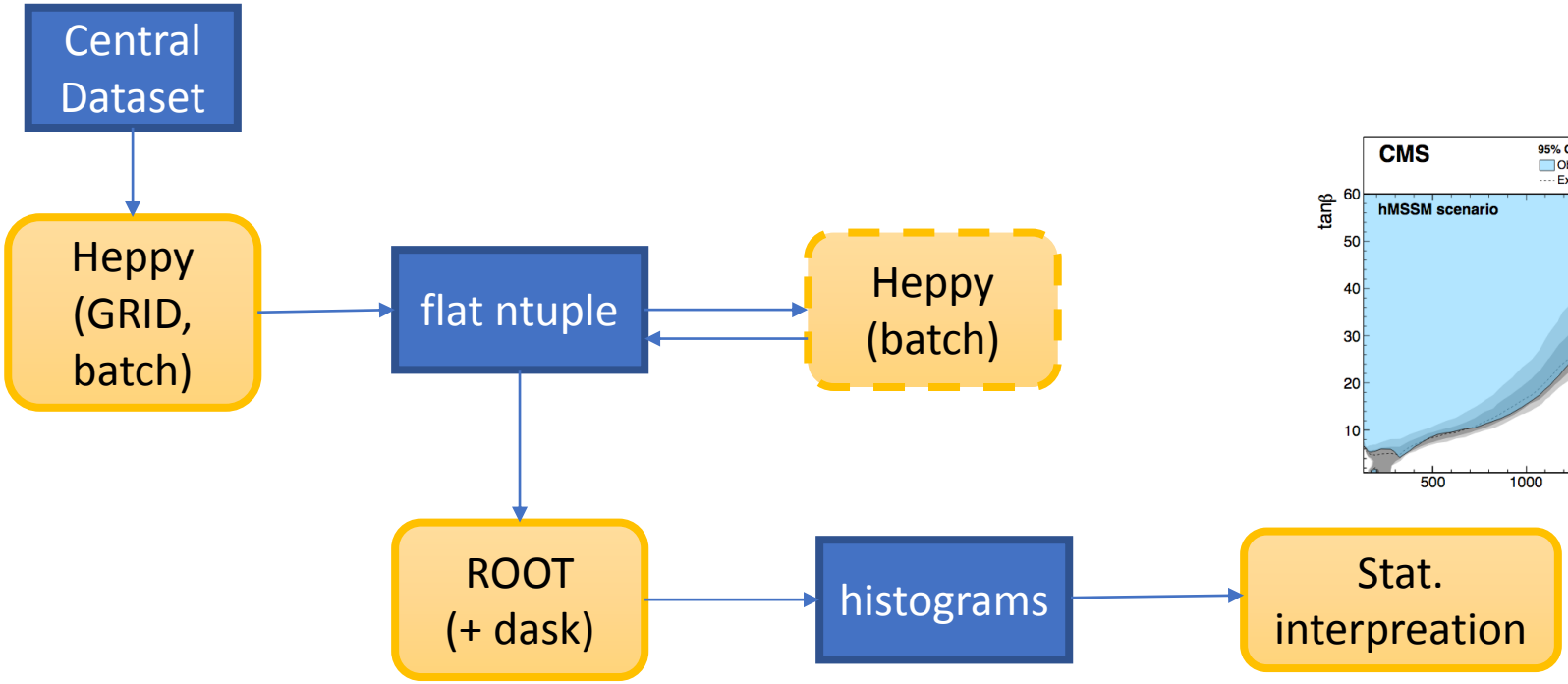
**Analyzers:**
- Fast & easy to write
- Can drive C++ (here fastjet)

**Objects:**
- Added to the event or modified in place
- Generic runtime dataformats provided

**Generic Analyzers:**
- Extremely flexible

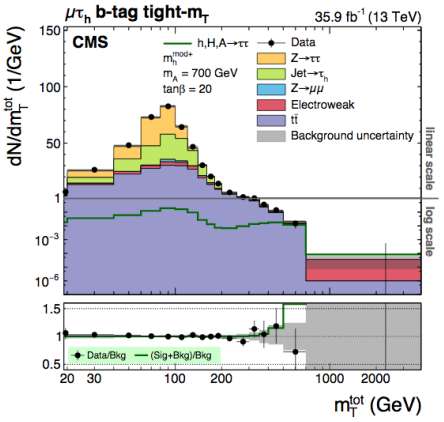
**Output:**
- Event or collection (e.g. jets)
- Ntuples, histograms, ndarrays, text, database entries, …

# Typical Analysis Workflow









# A Fully Generic Event Filter

```
from heppy.framework.analyzer import Analyzer

class EventFilter(Analyzer):

    def process(self, event):
        coll = getattr(event, self.cfg_ana.src)
        passed = self.cfg_ana.filter_func(coll)
        return passed
```

Full Analyzer Code

```
two_jets = AnalyzerCfg(
    EventFilter,
    src = 'jets'
    filter_func = lambda c: len(c)>1 and c[1].pt()>50.
    )
```

Analyzer Configuration

# Analysis Configuration

**Input Definition**

```
mc_drell_yan = Component(
    'drell_yan',
    files = ['dy_0.root', 'dy_1.root']
    split_factor = 2
    )

data_dimu = = Component(
    'drell_yan',
    files = ['dimu_0.root', 'dimu_1.root']
    split_factor = 1
    )

components = [mc_drell_yan, data_dimu]
```

2 tasks
(1 / file)

1 task

3 tasks:
- Multiprocessing
- Or batch, GRID

**Event Processing Sequence**

```
sequence = Sequence[
    event_source,
    muon_sequence,
    one_zed,
    jet_sequence,
    out_ntuple,
    out_pandas,
    ]
```

```
muon_sequence = Sequence(
    all_muons,
    muon_isolation,
    isolated_muons,
    two_muons
    )
```

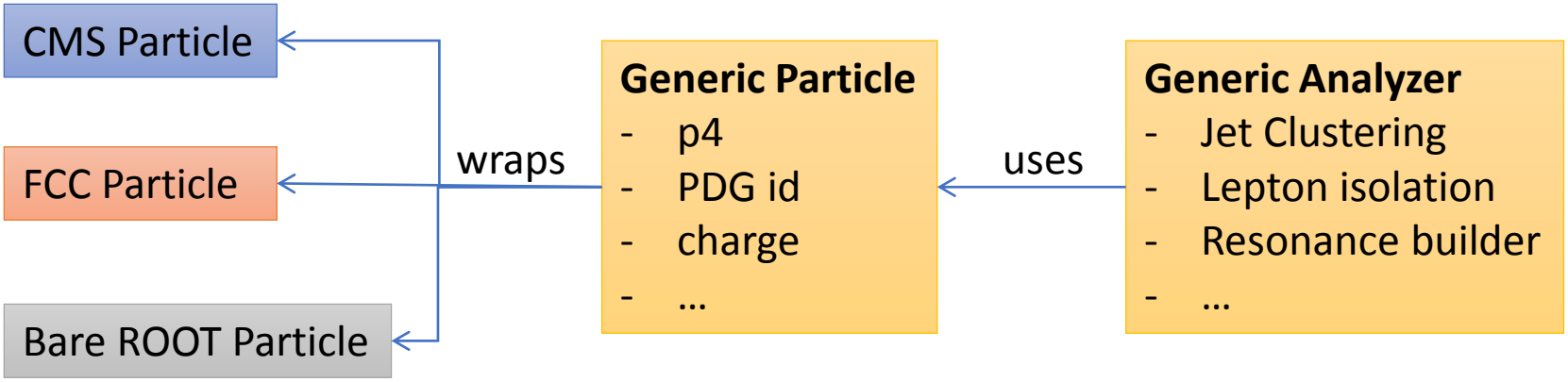
# Parallel processing

- Multiprocessing on local machine automatic
  - Interactive, single process, if only 1 task
    - testing, pdb, investigate particular events, event display
  - `heppy output_dir analysis_config.py`

- On the batch:
  - Several batch backends: LSF, HTCondor, Grid, nohup
  - 
```
heppy_batch.py -o output_dir \
               -b 'bsub -q 8nh < batchScript.sh' \
               analysis_config.py
```

# Generic Data Formats

- Analyzers can be used in different experiments
- How? Generic data formats



# Meta information tracking

- yaml files written to output directories

```
processing:
  nfiles:     100
  ngoodfiles: 99
sample:
  id: !!python/object:uuid.UUID
    int: 136575863457015515944380205888302330l6
  jobtype:    heppy
  mother:     pythia/ee_to_ZH_BS_Oct2
  nevents:    34948
  nfiles:     1
  ngoodfiles: 1
  pattern: fcc_ee_higgs.analyzers.LLWWTreeProducer.LLWWTreeProducer_1/tree.root
software:
  fcc_datasets:    !!python/unicode 'df41188443d8b15436543f854ee7de77a6934036'
  fcc_ee_higgs:    !!python/unicode '90c6d41342ff8c3b6fb35a4189f592c29ac4c5dd'
  heppy:           !!python/unicode 'deffa5922d02c43c1633e56c5f1928330d138c96'
```

all python packages with a git repo are tracked, even the private ones

# Users

- Standalone heppy
  - Includes CMS, ILC/CLIC, bare root backends
    - no dependency, install the needed software if you want to use it
  - 50 forks, ~25 analyses

- CMS heppy
  - Distributed in CMSSW
    - could move to standalone version
  - + CMS-specific Analyzers, data formats, tools
    (A. Rizzi, G. Petrucciani, CB, …)
  - > 170 forks, 50(?) analyses

# Installation

- https://pypi.org/project/heppyfwk/
  - https://github.com/cbernet/heppy/blob/master/doc/Heppy_-_Installation_Instructions.md

- Summary:
  - install anaconda for python 2.X (or have python 2.7)
  - `pip install heppyfwk`
  - + if you want to use ROOT:
    need to compile ROOT for your python

# Some answers:

- Analysis steps and evolution:
  - transform, select, and compress data
  - interactive debugging and understanding of single events → mass processing
  - systematics: run specific jobs with shifted parameters
- Analysis workflow:
  - can read any dataset from central datasets to reduced ntuples.
  - after heppy: e.g. root, pandas, dask
- Scaling:
  - GRID, batch clusters
- Missing functionalities / todo
  - cannot write out structured root files
    - only flat ntuples, numpy arrays, …
  - performance limited by reading input files (PyROOT)
  - need porting to python 3
- Preservation and sharing
  - share code with git
  - share configuration files
  - automatic software version tracking of all tools in a git repo (gitpython) at running time

# Heppy: Summary

- Modular sequential event processing framework
  - easy, flexible, re-usable, scalable
- Your own version of Gaudi, CMSSW or Marlin
  - transform, select, write out
  - the goal is not to make plots
- Used by 100 people
  - 50-100 published analyses