Fermilab

20 May 2026

CODE4HEP STATUS

Lindsey Gray
Scientist

U.S. DEPARTMENT of ENERGY
Fermi National Accelerator Laboratory is managed by FermiForward for the U.S. Department of Energy Office of Science

FERMI NATIONAL ACCELERATOR LABORATORY

CODE4HEP Justification

More than a decade of experience operating online and offline software at LHC

- A wealth of operational knowledge and well-honed patterns in intense data-taking
  - Particularly exploiting multi-core architectures and designing high throughput systems in that context
  - Heterogeneous architectures
  - For reconstruction and simulation, vast experience in understanding interplay of threading models
- FCCee and Muon Collider are similarly intense data-taking environments

CMSSW is a state-of-the-art multithreaded event processing framework

- Good performance of core framework across workloads, module implementation quality governs throughput, degree of thread safety encoded in inheritance patterns
- Well-patterned handling of detector conditions in multithreaded environment across all relevant time scales (e.g. run, luminosity section)

However: CMSSW does not speak EDM4HEP, making it difficult to practically demonstrate how this wealth of knowledge maps on to future collider software needs as to inform future design choices.

CODE4HEP Overview

The proof of concept:

- Bring EDM4HEP into fork of CMSSW, patch CMSSW to read and write EDM4HEP files
  - To work out the mismatches in conceptual design, and map matching parts well
  - This work is largely complete

Extract the core of CMSSW, “Stitched”, as a maintainable, standalone package

- In progress in parallel, with buildable modern examples available
  - *Eventually leading to wrapped key4hep modules running in stitched*
- Develop generator interfaces so that we may run consolidated workflows
  - This was waiting for the above, and community consensus on exchange formats, and is just starting
- Integrate and refine GEANT4 multithreading in a complete, multi-purpose framework
  - We are working with GEANT4 developers to harmonize / streamline its internal threading model
  - Evaluating this also in the context of BIB simulation and event mixing
  - In progress with useful insights to discuss

Finally, use modern analysis software stacks on the output of these products.

In this way we demonstrate interoperability at all levels, and contribute to wider FC software community while creating a testbed.

01

CODE4HEP Proof of Concept

Diving into the proof-of-concept!

- Reproducer available at: https://github.com/code4hep/build

Installation

```
mkdir -p scratch
cd scratch
git clone git@github.com:code4hep/build
cd build
./setup.sh
```

To enable debug symbols in all external and CMSSW builds, change the last line:

```
./setup.sh -d
```

```
Copying tmp/el9_amd64_gcc13/src/Code4hep/DataFormats/src/Code4hepDataFormats/libCode4hepDataFormats.so to productstore area:
>> Checking EDM Class Version for src/Code4hep/DataFormats/src/classes_def.xml in Code4hepDataFormats
@@@@ ----> OK EDM Class Version Code4hepDataFormats
>> Checking EDM Class Transients in Code4hepDataFormats
@@@@ ----> OK EDM Class Transients Code4hepDataFormats
Leaving library rule at Code4hep/DataFormats
>> Leaving Package Code4hep/DataFormats
>> Package Code4hep/DataFormats built
>> Subsystem Code4hep built
gmake[1]: Entering directory '/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1'
>> Local Products Rules ..... started
>> Local Products Rules ..... done
@@@@ Refreshing Plugins:edmPluginRefresh for lib/el9_amd64_gcc13
>> Creating project symlinks
>> Done python_symlink
>> Plugins of all types refreshed.
>> Compiling python3 modules cfipython/el9_amd64_gcc13
>> Compiling python3 modules python
>> Done generating edm plugin poisoned information
>> All python modules compiled
gmake[1]: Leaving directory '/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1'
lagray@cmslpc364:~/nobackup/code4hep/build$
```

Then go make a coffee…

At the end ->

```
lagray@cmslpc364:/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1/src/Code4hep/Test/test$ ./run_podioSource.sh
Writing frame 0
Writing frame 1
Writing frame 2
++ finished: construction of Services
++ finished: construction of Services
++ starting: construction of EventSetup modules
++ starting: construction of EventSetup modules
++ finished: construction of EventSetup modules
++ finished: construction of EventSetup modules
++ starting: constructing source: PodioSource
++ starting: constructing source: PodioSource
19-May-2026 12:41:12 CDT  Successfully opened file edm4hep.root
19-May-2026 12:41:12 CDT  Successfully opened file edm4hep.root
++ finished: constructing source: PodioSource
++ finished: constructing source: PodioSource
++++ starting: constructing module with label 'TriggerResults' id = 1
++++ starting: constructing module with label 'TriggerResults' id = 1
++++ finished: constructing module with label 'TriggerResults' id = 1
++++ finished: constructing module with label 'TriggerResults' id = 1
```

How'd we do this? (I)

```
[lagray@cmslpc364:/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1/src/Code4hep/DataFormats/src$ cat classes_def.xml
 <lcgdict>

 <class name="edm::Wrapper<edm4hep::CaloHitContributionCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::CalorimeterHitCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::ClusterCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::EventHeaderCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::GeneratorEventParametersCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::MCParticleCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::ParticleIDCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::RawCalorimeterHitCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::RawTimeSeriesCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::RecDqdxCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::ReconstructedParticleCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::SenseWireHitCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::SimCalorimeterHitCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::SimTrackerHitCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::TimeSeriesCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::TrackCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::TrackerHitPlaneCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::TrackerHit3DCollection>" persistent="false"/>
 <class name="edm::Wrapper<edm4hep::VertexCollection>" persistent="false"/>

 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::CalorimeterHit,edm4hep::MCParticle>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::CalorimeterHit,edm4hep::SimCalorimeterHit>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::Cluster,edm4hep::MCParticle>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::ReconstructedParticle,edm4hep::MCParticle>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::Track,edm4hep::MCParticle>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::TrackerHit,edm4hep::SimTrackerHit>>" persistent="false"/>
 <class name="edm::Wrapper<podio::LinkCollection<edm4hep::Vertex,edm4hep::ReconstructedParticle>>" persistent="false"/>

 <class name="edm::Wrapper<podio::UserDataCollection<int32_t>>" persistent="false"/>
 <class name="edm::Wrapper<podio::UserDataCollection<float>>" persistent="false"/>

 <class name="podio::LinkCollection<edm4hep::CalorimeterHit,edm4hep::MCParticle>"/>
 <class name="podio::LinkCollection<edm4hep::CalorimeterHit,edm4hep::SimCalorimeterHit>"/>
 <class name="podio::LinkCollection<edm4hep::Cluster,edm4hep::MCParticle>"/>
 <class name="podio::LinkCollection<edm4hep::ReconstructedParticle,edm4hep::MCParticle>"/>
 <class name="podio::LinkCollection<edm4hep::Track,edm4hep::MCParticle>"/>
 <class name="podio::LinkCollection<edm4hep::TrackerHit,edm4hep::SimTrackerHit>"/>
 <class name="podio::LinkCollection<edm4hep::Vertex,edm4hep::ReconstructedParticle>"/>

 <class name="podio::LinkT<edm4hep::CalorimeterHit,edm4hep::MCParticle, false>"/>
 <class name="podio::LinkT<edm4hep::CalorimeterHit,edm4hep::SimCalorimeterHit, false>"/>
 <class name="podio::LinkT<edm4hep::Cluster,edm4hep::MCParticle, false>"/>
 <class name="podio::LinkT<edm4hep::ReconstructedParticle,edm4hep::MCParticle, false>"/>
 <class name="podio::LinkT<edm4hep::Track,edm4hep::MCParticle, false>"/>
 <class name="podio::LinkT<edm4hep::TrackerHit,edm4hep::SimTrackerHit, false>"/>
 <class name="podio::LinkT<edm4hep::Vertex,edm4hep::ReconstructedParticle, false>"/>

 </lcgdict>
```

- Exposed podio and edm4hep classes to Stitched
- Created a translation layer from Stitched semantics to edm4hep file schema
  - Occasionally large differences in how edm4hep labels event data products compared to Stiched
- Developed output module to write podio frames to root files
- To fix: crash for links in certain cases

```
lagray@cmslpc364:/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1/src/Code4hep/IO/plugins$ ls -ltrh
total 60K
-rw-r--r-- 1 lagray us_cms  219 May 19 12:15 BuildFile.xml
-rw-r--r-- 1 lagray us_cms 6.8K May 19 12:15 PodioFile.cc
-rw-r--r-- 1 lagray us_cms 2.3K May 19 12:15 PodioFile.h
-rw-r--r-- 1 lagray us_cms 4.6K May 19 12:15 PodioOutputModule.cc
-rw-r--r-- 1 lagray us_cms 6.0K May 19 12:15 PodioSource.cc
-rw-r--r-- 1 lagray us_cms  581 May 19 12:15 TypeNameConversion.cc
-rw-r--r-- 1 lagray us_cms  450 May 19 12:15 TypeNameConversion.h
-rw-r--r-- 1 lagray us_cms 4.7K May 19 12:15 fillProductRegistry.cc
-rw-r--r-- 1 lagray us_cms  721 May 19 12:15 fillProductRegistry.h
-rw-r--r-- 1 lagray us_cms 2.9K May 19 12:15 putOnReadForAllProducts.cc
-rw-r--r-- 1 lagray us_cms  720 May 19 12:15 putOnReadForAllProducts.h
lagray@cmslpc364:/uscms_data/d2/lagray/code4hep/CMSSW_16_1_0_pre1/src/Code4hep/IO/plugins$
```

How'd we do this? (II)

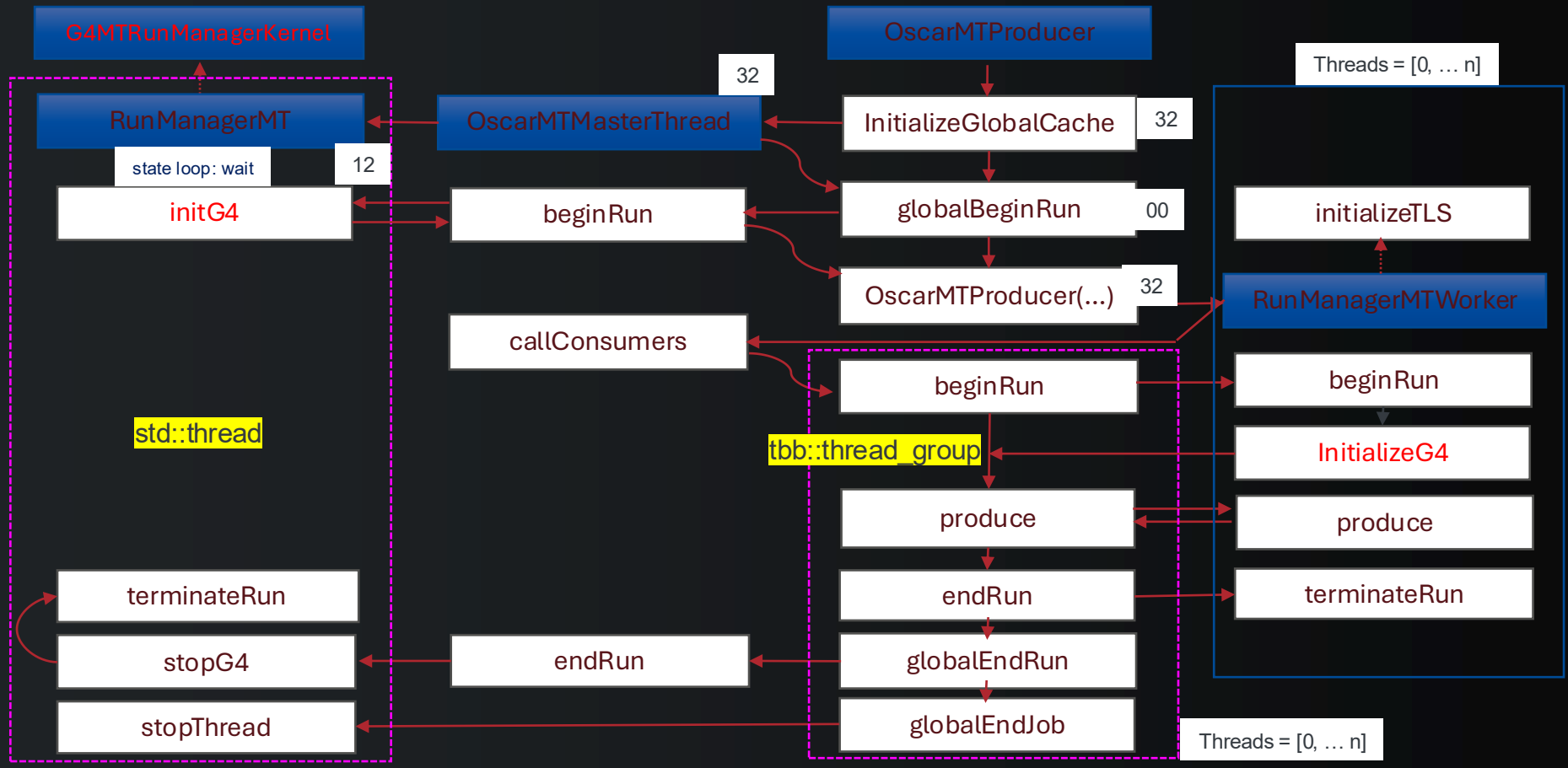
```
void PodioOutputModule::write(edm::EventForOutput const& e) {
  auto factory = code4hep::CollectionWrapperConverterBaseFactory::get();

  bool haveEventHeader = false;
  podio::Frame frame;
  for (auto const& product : keptProducts()[edm::InEvent]) {
    auto const& typeName = typeNameConversion(product.first->fullClassName());

    if (product.first->moduleLabel() == "EventHeader") {
      haveEventHeader = true;
    }
    auto converter = factory->create(typeName);

    edm::TypeID const& tid = product.first->unwrappedTypeID();
    edm::EDGetToken const& token = product.second;
    edm::BasicHandle bh = e.getByToken(token, tid);

    std::string collectionName = product.first->moduleLabel() + product.first->productInstanceName();
    if (bh.isValid()) {
      assert(bh.wrapper());
      auto collectionBase = converter->getCollection(*bh.wrapper());
      assert(collectionBase);
      auto copiedCollection = converter->copy(*const_cast<podio::CollectionBase*>(collectionBase));
      assert(copiedCollection);
      frame.put(std::move(copiedCollection), collectionName);
    } else {
      //Data product is missing, but we MUST include one
      auto empty = converter->createEmpty();
      frame.put(std::move(empty), collectionName);
    }
  }
  if (not haveEventHeader) {
    auto ehc = std::make_unique<edm4hep::EventHeaderCollection>();
    auto eh = ehc->create();
    eh.setEventNumber(e.id().event());
    eh.setRunNumber(e.id().run());
    eh.setTimeStamp(e.time().value());
    frame.put(std::move(ehc), "EventHeader");
  }
  writer_.writeEvent(frame);
}
```

W. Dagenhart, C. Jones, K. Pedro

02

GEANT4 Development Work

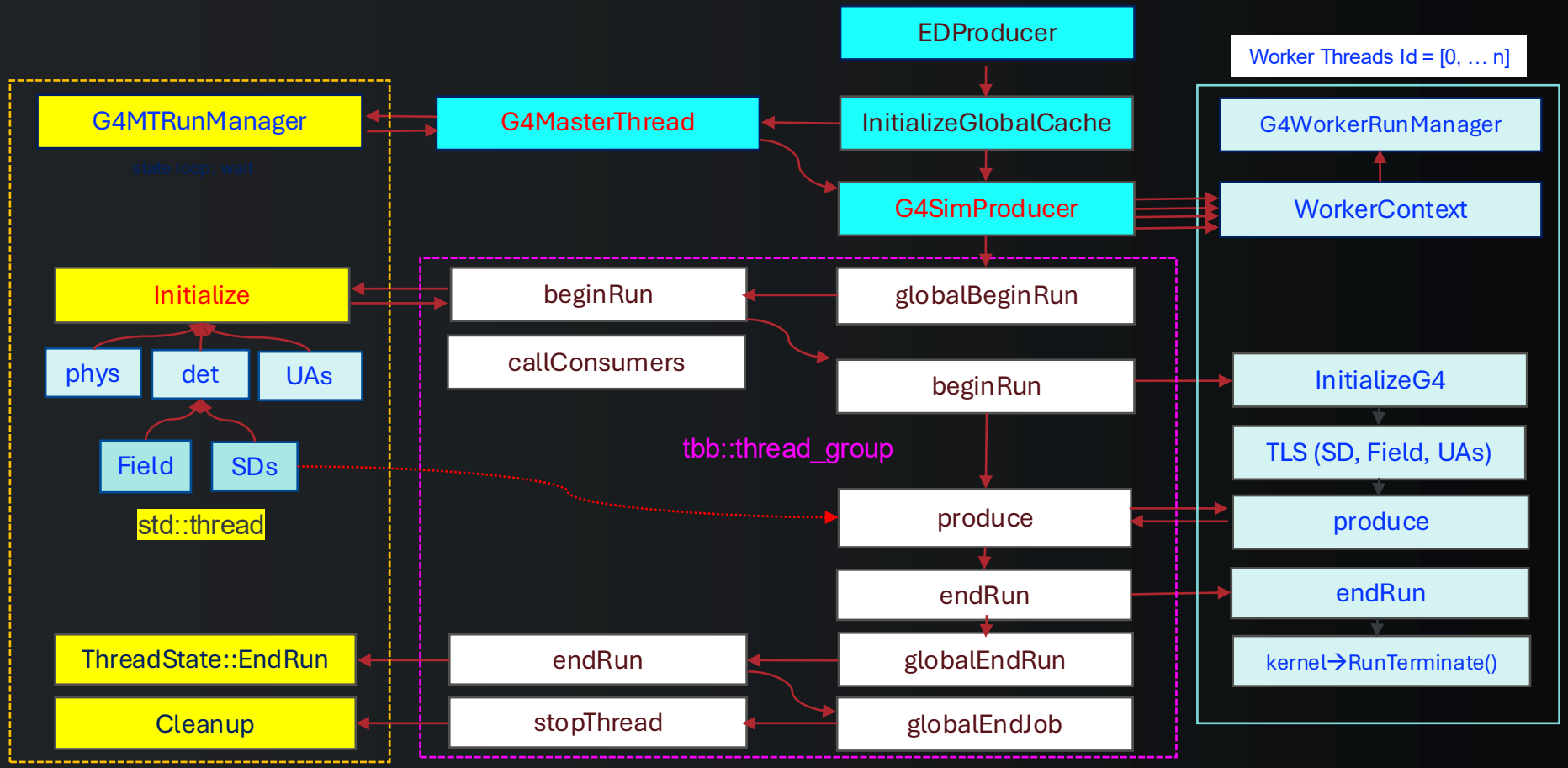
OscarMTProducer and Geant4: Two Thread Models



Large migration to using immutable shared data and thread local data where appropriate.

C. Jones, S. Jun, M. Kortelainen

Refactored: more correspondence between threads



Large migration to using immutable shared data and thread local data where appropriate.

C. Jones, S. Jun, M. Kortelainen

Geant4 Refactorization Impact

- Multithreading now available at the coarsest boundaries of conditions data (Runs)
  - More efficient use of cores when running out of events to process during run-dependent MC production
  - This is likely quite pertinent for FCC where conditions will be following ~5-minute-long spills over multiple hour detector runs
- This refactorization could be used as implementation by any other framework
  - The architecture is specific to CMSSW **but** the way that threads are handled could be used in any event processing framework
  - Improvements to multithreading data architecture in GEANT4 shared by all client code
- Likely benefits to FCCee and Muon Collider simulation efforts
  - Muon Collider can benefit from better sharing of memory from event to event in BIB simulation where each event is very complex (worse than HL-LHC in total particle flux)
  - FCCee benefits from being able to scale horizontally for less complex beam background and hard scatter events

03

Stitched and Analysis

Stitched: The Next Generation

- New extraction of CMSSW core framework being developed at:
  - https://github.com/code4hep/stitched-alpha2
- Issues around contribution and code ownership being considered:
  - Complete git and cvs history of CMSSW (back to CMSSW_0_0_0!) are saved in this extraction
  - Scripts for fetching upstream CMSSW core and performing complete extraction are available and being automated
  - Spack integration for easy installation with alongside key4hep already worked out
    - Initial iterations of stitched were already ensuring this
  - Current status is such that you can do this:
    - And get a functioning cmsRun
  - Should be able soon to take code4hep proof of concept and run it entirely within stitched instead of CMSSW
    - Achieve experiment agnostic framework that has decades of operational time behind its design.

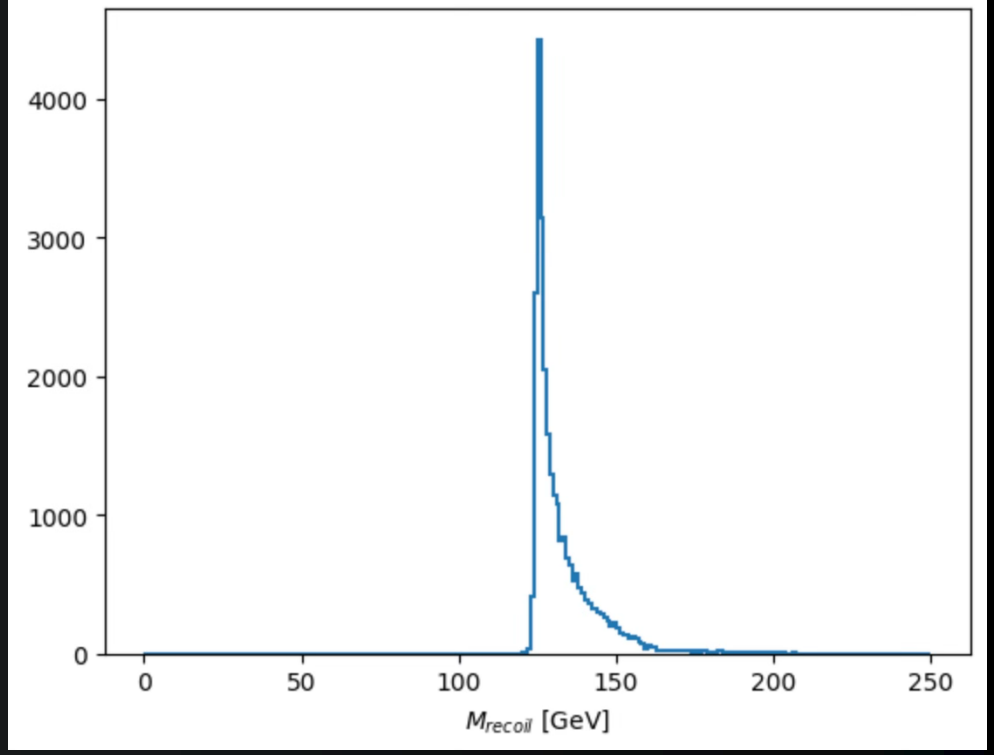
```
git clone --depth=2 https://github.com/spack/spack.git
source spack/share/spack/setup-env.sh
spack compiler find
spack install gcc@14 # because LPC heavy01 system gcc is 8.5
spack repo add https://github.com/code4hep/c4h-spack-packages
spack install stitched cxxstd=20 %gcc@14 %root@6.36
spack load stitched
source $(which stitched_env.sh)
```

- Next steps include extracting heterogeneous compute interface from CMSSW
  - e.g.: SONIC, alpaca, etc., along with asynchronous, non-blocking interfaces to each for maximum efficiency

Interface to Analysis

```
from coffea.nanoevents import NanoEventsFactory, EDM4HEPSchema
import dask_awkward as dak
import hist.dask as hda
import numpy as np

events = NanoEventsFactory.from_root(
    "~/Downloads/rv02-02.sv02-02.mILD_l5_o1_v02.E250-SetA.I402004"
    ".Pe2e2h.eR.pL.n000.d_dstm_15090_*.slcio.edm4hep.root",
    treepath="events",
    schemaclass=EDM4HEPSchema,
    permit_dask=True,
).events()

E_cm = 250.0
Z_pole = 91.1876

muons = events.PandoraPFOs[abs(events.PandoraPFOs.pdgId) == 13]
dimuons = dak.combinations(muons, 2, fields=["mu1", "mu2"])
os_dimuons = dimuons[dimuons.mu1.charge == -dimuons.mu2.charge]
os_p4 = os_dimuons.mu1 + os_dimuons.mu2
nearest_zpole = os_p4[
    dak.singletons(dak.argmin(abs(os_p4.mass2 - Z_pole**2), axis=1))
]

recoil = np.sqrt(E_cm**2 + nearest_zpole.mass2 - 2 * nearest_zpole.energy * E_cm)

recoil_hist = (
    hda.Hist.new.Reg(250, 0, 250, name="recoil", label="$M_{recoil}$ [GeV]")
    .Double()
    .fill(dak.flatten(recoil, axis=1))
)
recoil_hist.compute().plot1d()
dak.necessary_columns(recoil_hist)
```

```
{'from-uproot-3dccf8a319bc10377dc7c72a088c60b1': ['PandoraPFOs.t',
  'PandoraPFOs.z',
  'PandoraPFOs.charge',
  'PandoraPFOs.y',
  'PandoraPFOs.pdgId',
  'PandoraPFOs.x']}
```



- Unfortunately, we do not have a presently working demonstration of running coffea on edm4hep output from CMSSW, the file format has changed too much since initial development of the schema.
- However, we would like to return to the state demonstrated above.

Remarks / Conclusions

- Large degree of work complete for interfacing edm4hep with CMSSW
  - Can read and write test data using actual podio classes, test close on edm4hep data produced by CMSSW
- GEANT4 refactoring work proceeding well, with benefits for any multithreaded framework that uses GEANT
- CMSSW extracted as “stitched” and is in a mature state of development
  - Soon hope to move to using stitched as opposed to edm4hep-grafted version of CMSSW
- Data analysis unfortunately lost to version skew – but working with authors of edm4hep interpretation in coffea to return this to functionality
  - Lots of work has been going on in edm4hep!