Example analysis with Fun4All/Evaluators

SIDIS WG meeting
May 12, 2022

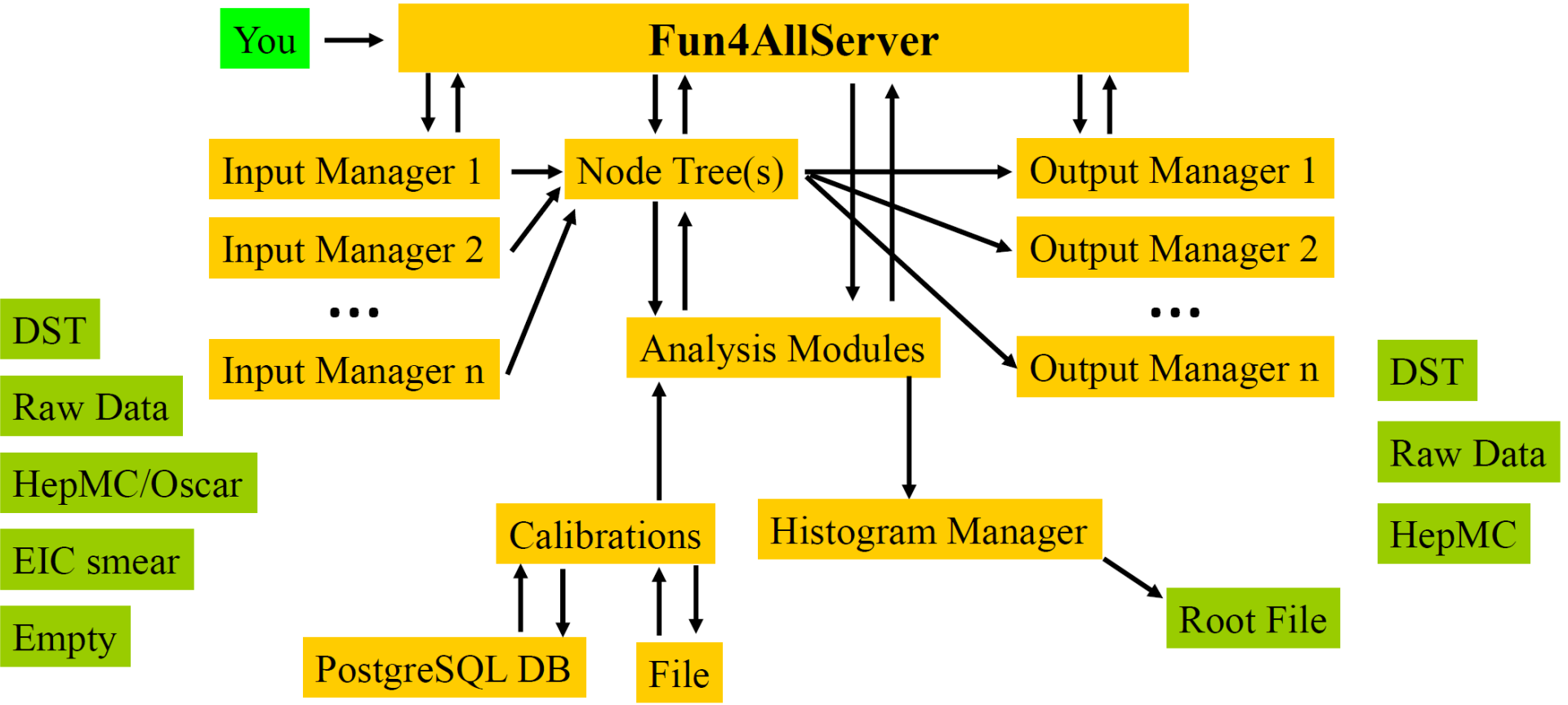
Ralf Seidl (RIKEN)

General resources on Fun4All

- Chris Pinkenburgs tutorial: https://indico.bnl.gov/event/11112/contributions/47257/attachments/33390/53662/Fun4ECCE.pdf
- Joe Osborn's analysis tutorial: https://indico.bnl.gov/event/11112/contributions/47258/attachments/33410/53691/AnaTutorial.pdf

RIKEN

Fun4All Framework in a nutshell



That's all there is to it, no backdoor communications – steered by ROOT macros

Slide from Chris Pinkenburg

The Node Tree

```
-------------------------------------

List of Nodes in Fun4AllServer:
Node Tree under TopNode TOP
TOP (PHCompositeNode)/
   DST (PHCompositeNode)/
      PHHepMCGenEventMap (IO,PHHepMCGenEventMap)
      Sync (IO,SyncObjectv1)
      EventHeader (IO,EventHeaderv1)
      G4HIT_BH_1 (IO,PHG4HitContainer)
      G4TruthInfo (IO,PHG4TruthInfoContainer)
      MVTX (PHCompositeNode)/
         G4HIT_MVTX (IO,PHG4HitContainer)
      INTT (PHCompositeNode)/
         G4HIT_INTT (IO,PHG4HitContainer)
      TPC (PHCompositeNode)/
         G4HIT_TPC (IO,PHG4HitContainer)
         G4HIT_ABSORBER_TPC (IO,PHG4HitContainer)

.............
```

- The node tree is where all of the data is stored in any Fun4All job
- Users interact with the node tree to analyze, create, manipulate, data that they are interested in
- Nodes are accessed by asking the node tree

```
/// Get the reconstructed tower jets
JetMap *reco_jets = findNode::getClass<JetMap>(topNode, "AntiKt_Tower_r04");
/// Get the truth jets
JetMap *truth_jets = findNode::getClass<JetMap>(topNode, "AntiKt_Truth_r04");
```

Object type (JetMap) | Node tree to search (topNode) | Node name on node tree (AntiKt_Truth_r04)

Nodes are Not TTrees!

Slide from Joe Osborn

RIKEN

Analysis Module

- Analysis modules *must* inherit from SubsysReco base class. Tells Fun4All how to treat it
- Several methods called by Fun4All:
  - `Init(PHCompositeNode *topNode)`
  - `InitRun(PHCompositeNode *topNode)`
  - `process_event(PHCompositeNode *topNode)`
  - `ResetEvent(PHCompositeNode *topNode)`
  - `EndRun(const int runnumber)`
  - `End(PHCompositeNode *topNode)`
- Each houses the analysis code that you want to run at a given time in processing (initially, for each event, and at the end of the job)
- Take advantage of existing infrastructure, e.g. `CreateSubsysRecoModule.pl <Module Name>`

```
class AnaTutorial : public SubsysReco
{
 public:
  /// Constructor
  AnaTutorial(const std::string &name = "AnaTutorial",
              const std::string &fname = "AnaTutorial.root");

  // Destructor
  virtual ~AnaTutorial();

  /// SubsysReco initialize processing method
  int Init(PHCompositeNode *);

  /// SubsysReco event processing method
  int process_event(PHCompositeNode *);

  /// SubsysReco end processing method
  int End(PHCompositeNode *);
```

General resources on EventEvaluator

- https://ecce-eic.github.io/
- Lates EventEvaluator file: https://github.com/ECCE-EIC/fun4all_eicdetectors/blob/master/analysis/eiceevaluator/EventEvaluatorEIC.cc
- Doxygen information: https://ecce-eic.github.io/doxygen/d7/ddc/EventEvaluatorEIC_8cc.html

Main Methods of the EventEvaluator

```
8  //============================================
9
10 #include <fun4all/SubsysReco.h>
11
12 #include <set>
13 #include <string>
14 #include <vector>
15
16 class CaloEvalStack;
17 class PHCompositeNode;
18 class PHHepMCGenEventMap;
19 class PHHepMCGenEvent;
20 class TFile;
21 class TNtuple;
22 class TTree;  //Added by Barak
23
32 class EventEvaluatorEIC : public SubsysReco
33 {
34  public:
35   enum class TrackSource_t : unsigned short
36   {
37     all = 0,
38     inner = 1,
39     silicon = 2,
40     ttl = 3
41   };
42
43   EventEvaluatorEIC(const std::string& name = "EventEvaluatorEIC",
44                     const std::string& filename = "g4eval_cemc.root");
45   ~EventEvaluatorEIC() override{};
46
47   int Init(PHCompositeNode* topNode) override;
48   int process_event(PHCompositeNode* topNode) override;
49   int End(PHCompositeNode* topNode) override;
50
51   void set_strict(bool b) { _strict = b; }
52
53   void set_do_store_event_level_info(bool b) { _do_store_event_info = b; }
```

Initialization (create tree, add tracks to it)

```
int EventEvaluatorEIC::Init(PHCompositeNode* topNode)
{
  _ievent = 0;

  _tfile = new TFile(_filename.c_str(), "RECREATE");

  _event_tree = new TTree("event_tree", "event_tree");
```

```
if (_do_TRACKS)
{
  _event_tree->Branch("nTracks", &_nTracks, "nTracks/I");
  _event_tree->Branch("tracks_ID", _track_ID, "tracks_ID[nTracks]/F");
  _event_tree->Branch("tracks_charge", _track_charge, "tracks_charge[nTracks]/S");
  _event_tree->Branch("tracks_px", _track_px, "tracks_px[nTracks]/F");
  _event_tree->Branch("tracks_py", _track_py, "tracks_py[nTracks]/F");
  _event_tree->Branch("tracks_pz", _track_pz, "tracks_pz[nTracks]/F");
  _event_tree->Branch("tracks_x", _track_x, "tracks_x[nTracks]/F");
  _event_tree->Branch("tracks_y", _track_y, "tracks_y[nTracks]/F");
  _event_tree->Branch("tracks_z", _track_z, "tracks_z[nTracks]/F");
  _event_tree->Branch("tracks_ndf", _track_ndf, "tracks_ndf[nTracks]/F");
  _event_tree->Branch("tracks_chi2", _track_chi2, "tracks_chi2[nTracks]/F");
  _event_tree->Branch("tracks_dca", _track_dca, "tracks_dca[nTracks]/F");
  _event_tree->Branch("tracks_dca_2d", _track_dca_2d, "tracks_dca_2d[nTracks]/F");
  _event_tree->Branch("tracks_trueID", _track_trueID, "tracks_trueID[nTracks]/F");
  _event_tree->Branch("tracks_source", _track_source, "tracks_source[nTracks]/s");

  if (_do_PID_LogLikelihood)
```

Getting several nodes (here FHCAL Towers and Clusters)

```
int EventEvaluatorEIC::process_event(PHCompositeNode* topNode)
{
  if (Verbosity() > 0)
  {
    cout << "entered process_event" << endl;
  }
  if (_do_FHCAL)
  {
    if (!_caloevalstackFHCAL)
    {
      _caloevalstackFHCAL = new CaloEvalStack(topNode, "FHCAL");
      _caloevalstackFHCAL->set_strict(_strict);
      _caloevalstackFHCAL->set_verbosity(Verbosity() + 1);
    }
    else
    {
      _caloevalstackFHCAL->next_event(topNode);
    }
  }
```

Tracking information

```
if (_do_TRACKS)
{
  _nTracks = 0;
  _nProjections = 0;

  EICPIDParticleContainer* pidcontainer(nullptr);

  if (_do_PID_LogLikelihood)
  {
    pidcontainer = findNode::getClass<EICPIDParticleContainer>(topNode, "EICPIDParticleMap");
    if (pidcontainer == nullptr)
    {
      cout << __PRETTY_FUNCTION__ << " Error: missing EICPIDParticleMap while _do_PID_LogLikelihood = "
           << _do_PID_LogLikelihood << endl;
    }
  }

  // Loop over track maps, identifiy each source.
  // Although this configuration is fixed here, it doesn't require multiple sources.
  // It will only store them if they're available.
  std::vector<std::pair<std::string, TrackSource_t>> trackMapPairs = {
      {"TrackMap", TrackSource_t::all},
      {"InnerTrackMap", TrackSource_t::inner},
      {"SiliconTrackMap", TrackSource_t::silicon},
      {"TTLTrackMap", TrackSource_t::ttl}
  };
  bool foundAtLeastOneTrackSource = false;
  for (const auto& trackMapInfo : trackMapPairs)
  {
    if (_nTracks >= _maxNTracks) break;

    SvtxTrackMap* trackmap = findNode::getClass<SvtxTrackMap>(topNode, trackMapInfo.first);
    if (trackmap)
    {
```

Tracking info continued

```
for (SvtxTrackMap::ConstIter track_itr = trackmap->begin(); track_itr != trackmap->end(); track_itr++)
{
  if (_nTracks >= _maxNTracks) break;

  SvtxTrack_FastSim* track = dynamic_cast<SvtxTrack_FastSim*>(track_itr->second);
  if (track)
  {
    _track_ID[_nTracks] = track->get_id();
    _track_charge[_nTracks] = track->get_charge();
    _track_px[_nTracks] = track->get_px();
    _track_py[_nTracks] = track->get_py();
    _track_pz[_nTracks] = track->get_pz();
    _track_x[_nTracks] = track->get_x();
    _track_y[_nTracks] = track->get_y();
    _track_z[_nTracks] = track->get_z();
    _track_ndf[_nTracks] = track->get_ndf();
    _track_chi2[_nTracks] = track->get_chisq();
    // Ideally, would be dca3d_xy and dca3d_z, but these don't seem to be calculated properly in the
    // current (June 2021) simulations (they return NaN). So we take dca (seems to be ~ the 3d distance)
    // and dca_2d (seems to be ~ the distance in the transverse plane).
    // The names of the branches are based on the method names.
    _track_dca[_nTracks] = static_cast<float>(track->get_dca());
    _track_dca_2d[_nTracks] = static_cast<float>(track->get_dca2d());
    _track_trueID[_nTracks] = track->get_truth_track_id();
    _track_source[_nTracks] = static_cast<unsigned short>(trackMapInfo.second);
    if (_do_PROJECTIONS)
    {
      // find projections
      for (SvtxTrack::ConstStateIter trkstates = track->begin_states(); trkstates != track->end_states(); ++trkstates
      {
```

G4Truthparticles (includes secondaries, etc)

```
if (_do_MCPARTICLES)
{
  PHG4TruthInfoContainer* truthinfocontainer = findNode::getClass<PHG4TruthInfoContainer>(topNode, "G4TruthInfo");
  if (truthinfocontainer)
  {
    if (Verbosity() > 0)
    {
      cout << "saving MC particles" << endl;
    }
    //GetParticleRange for all particles
    //GetPrimaryParticleRange for primary particles
    PHG4TruthInfoContainer::ConstRange range = truthinfocontainer->GetParticleRange();
    for (PHG4TruthInfoContainer::ConstIterator truth_itr = range.first; truth_itr != range.second; ++truth_itr)
    {
      PHG4Particle* g4particle = truth_itr->second;
      if (!g4particle) continue;

      int mcSteps = 0;
      PHG4Particle* g4particleMother = truth_itr->second;
```

```
        _mcpart_ID[_nMCPart] = g4particle->get_track_id();
        _mcpart_ID_parent[_nMCPart] = g4particle->get_parent_id();
        _mcpart_PDG[_nMCPart] = g4particle->get_pid();
        _mcpart_E[_nMCPart] = g4particle->get_e();
        _mcpart_px[_nMCPart] = g4particle->get_px();
        _mcpart_py[_nMCPart] = g4particle->get_py();
        _mcpart_pz[_nMCPart] = g4particle->get_pz();
        PHG4VtxPoint* vtxtmp = truthinfocontainer->GetVtx(g4particle->get_vtx_id());
        if (vtxtmp)
        {
          _mcpart_x[_nMCPart] = vtxtmp->get_x();
          _mcpart_y[_nMCPart] = vtxtmp->get_y();
          _mcpart_z[_nMCPart] = vtxtmp->get_z();
        }
        //BCID added for G4Particle --  HEPMC particle matching
        _mcpart_BCID[_nMCPart] = g4particle->get_barcode();
        // TVector3 projvec(_mcpart_px[0],_mcpart_py[0],_mcpart_pz[0]);
        // float projeta = projvec.Eta();
        _nMCPart++;
      }
```


RIKEN

HEPMC information (generator info)

```
_nHepmcp = 0;
if (_do_HEPMC)
{
  PHHepMCGenEventMap* hepmceventmap = findNode::getClass<PHHepMCGenEventMap>(topNode, "PHHepMCGenEventMap");
  if (hepmceventmap)
  {
    if (Verbosity() > 0)
    {
      cout << "saving HepMC output" << endl;
    }
    if (Verbosity() > 0)
    {
      hepmceventmap->Print();
    }

    for (PHHepMCGenEventMap::ConstIter eventIter = hepmceventmap->begin();
         eventIter != hepmceventmap->end();
         ++eventIter)
    {
      PHHepMCGenEvent* hepmcevent = eventIter->second;

        _hepmcp_x1 = pdfinfo->x1();
        _hepmcp_x2 = pdfinfo->x2();
        _hepmcp_Q2 = pdfinfo->scalePDF();

        // m_mpi = truthevent->mpi();

        _hepmcp_procid = truthevent->signal_process_id();

        if (Verbosity() > 2)
        {
          cout << " Iterating over an event" << endl;
        }
        for (HepMC::GenEvent::particle_const_iterator iter = truthevent->particles_begin();
             iter != truthevent->particles_end();
             ++iter)
        {
          _hepmcp_E[_nHepmcp] = (*iter)->momentum().e();
          _hepmcp_PDG[_nHepmcp] = (*iter)->pdg_id();
          _hepmcp_px[_nHepmcp] = (*iter)->momentum().px();
          _hepmcp_py[_nHepmcp] = (*iter)->momentum().py();
          _hepmcp_pz[_nHepmcp] = (*iter)->momentum().pz();
          _hepmcp_status[_nHepmcp] = (*iter)->status();
          _hepmcp_BCID[_nHepmcp] = (*iter)->barcode();
          _hepmcp_m2[_nHepmcp] = 0;
          _hepmcp_m1[_nHepmcp] = 0;
          if ((*iter)->production_vertex())
          {
            for (HepMC::GenVertex::particle_iterator mother = (*iter)->production_vertex()->particles_begin(HepMC::parents);
                 mother != (*iter)->production_vertex()->particles_end(HepMC::parents);
                 ++mother)
            {
              _hepmcp_m2[_nHepmcp] = (*mother)->barcode();
              if (_hepmcp_m1[_nHepmcp] == 0)
                _hepmcp_m1[_nHepmcp] = (*mother)->barcode();
            }
          }
```

DST → Evaluator

```
int Fun4All_G4_EICDetector(    const int nEvents = 1,
const string &inputFile = "testdst.root",    const
string &outputFile = "G4EICDetector.root",    const
string &embed_input_file = "",
//https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX
/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
const int skip = 0,    const string &outdir = ".")

{

…

 bool use_event_evaluator = true;

 Input::READHITS = true;

 INPUTREADHITS::filename[0] = inputFile;

…

 Enable::TRACKING = true;

 Enable::TRACKING_EVAL = Enable::TRACKING && true;

 if (use_event_evaluator)

  {

    EventEvaluator *eval = new
EventEvaluator("EVENTEVALUATOR",  outputroot +
"_eventtree.root");

    eval->set_reco_tracing_energy_threshold(0.05);

    eval->Verbosity(0);
```

```
if (Enable::TRACKING_EVAL)
    {

      eval->set_do_TRACKS(true);

      //eval->set_do_HITS(true); //Potential problem

      eval->set_do_PROJECTIONS(true);

      if (G4TRACKING::DISPLACED_VERTEX)

        eval->set_do_VERTEX(true);

     }

 //if (Enable::CEMC_EVAL) eval->set_do_CEMC(true);

    if (Enable::EEMC_EVAL) eval->set_do_EEMC(true);

 //if (Enable::FEMC_EVAL) eval->set_do_FEMC(true);

    if (Enable::HCALIN_EVAL) eval->set_do_HCALIN(true);

    if (Enable::HCALOUT_EVAL) eval->set_do_HCALOUT(true);

    if (Enable::FHCAL_EVAL) eval->set_do_FHCAL(true);

    //if (Enable::FHCAL_EVAL || Enable::FEMC_EVAL ||
Enable::EEMC_EVAL)

    ///  eval->set_do_CLUSTERS(true);

    eval->set_do_MCPARTICLES(true);

    eval->set_do_HEPMC(true);

    se->registerSubsystem(eval);

  }
```


RIKEN

EventEvaluator output

```
 TTree          *fChain;   //!pointer to the analyzed TTree or TChain
   Int_t           fCurrent; //!current Tree number in a TChain
// Fixed size dimensions of array or collections stored in the TTree if any.
   // Declaration of leaf types
   Int_t           nTracks;
   Float_t         tracks_ID[40];   //[nTracks]
   Float_t         tracks_px[40];   //[nTracks]
   Float_t         tracks_py[40];   //[nTracks]
   Float_t         tracks_pz[40];   //[nTracks]
   Float_t         tracks_trueID[40];   //[nTracks]
Int_t           nMCPart;
   Float_t         mcpart_ID[7000];   //[nMCPart]
   Float_t         mcpart_ID_parent[7000];   //[nMCPart]
   Float_t         mcpart_PDG[7000];   //[nMCPart]
   Float_t         mcpart_E[7000];   //[nMCPart]
   Float_t         mcpart_px[7000];   //[nMCPart]
   Float_t         mcpart_py[7000];   //[nMCPart]
   Float_t         mcpart_pz[7000];   //[nMCPart]
 Int_t           nHepmcp;
   Int_t           hepmcp_procid;
   Float_t         hepmcp_x1;
   Float_t         hepmcp_x2;
   Float_t         hepmcp_ID[150];   //[nHepmcp]
   Float_t         hepmcp_ID_parent[150];   //[nHepmcp]
   Float_t         hepmcp_status[150];   //[nHepmcp]
   Float_t         hepmcp_PDG[150];   //[nHepmcp]
   Float_t         hepmcp_E[150];   //[nHepmcp]
   Float_t         hepmcp_px[150];   //[nHepmcp]
   Float_t         hepmcp_py[150];   //[nHepmcp]
   Float_t         hepmcp_pz[150];   //[nHepmcp]
   Int_t           hepmcp_m1[150];   //[nHepmcp]
   Int_t           hepmcp_m2[150];   //[nHepmcp]
```

SvtxTrackMap node

- Relates to trueID from PHG4TruthInfoContainer

PHG4TruthInfoContainer node → PHG4Particle:

- Only final state/secondary particles
- Neg ID: secondary
- ancestry

HEPMC output (not yet in official EE):

- Process ID (Pythia 99, 131/132, 135/136, etc)
- All generated particles: use status (4/3/0/1) for initial/PS/decaying/final info (not Pythia!!!)
- Parton info

Various other evaluators not shown here (Calorimeters)

Summary

- EventEvaluator pulls the relevant information from all various nodes (Hits, Tracks, Projections, each Calorimeter Towers, Clusters, G4Particles, HEPMC particles)
- Simple event based TTree output

- All (SI)DIS Kinematics calculations, Analysis have been performed on the TTrees