Dual Phase light simulation in LArSoft

Simulation of the Light production in the GAr phase.

José Soto, Michel Sorel, Beatriz Tapia

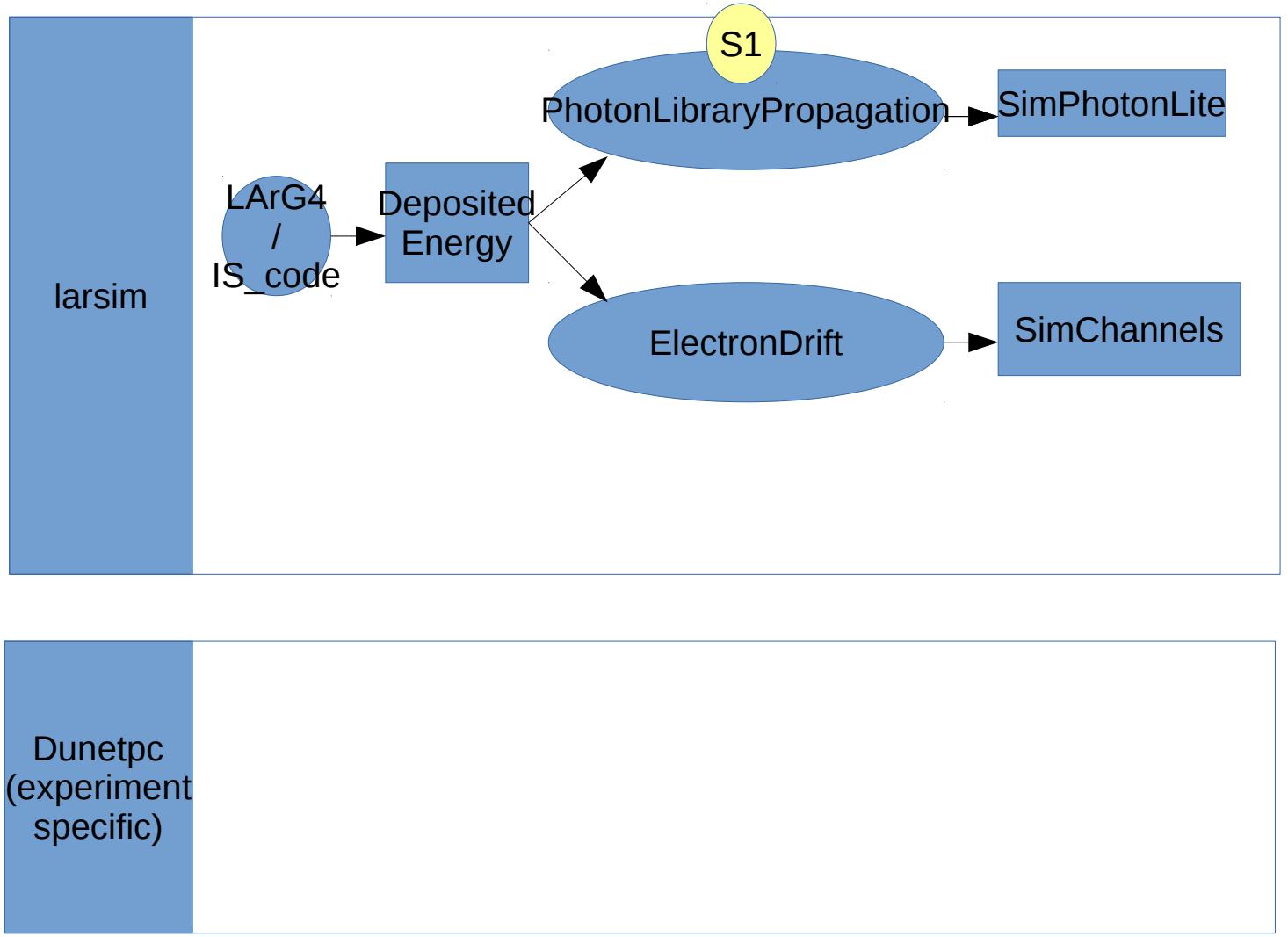
Goal of this talk

- To propose a design of the needed changes in the common repositories of larsoft to accommodate the simulation of the S2 light (light produced in the gas phase by the drifted electrons).

Content

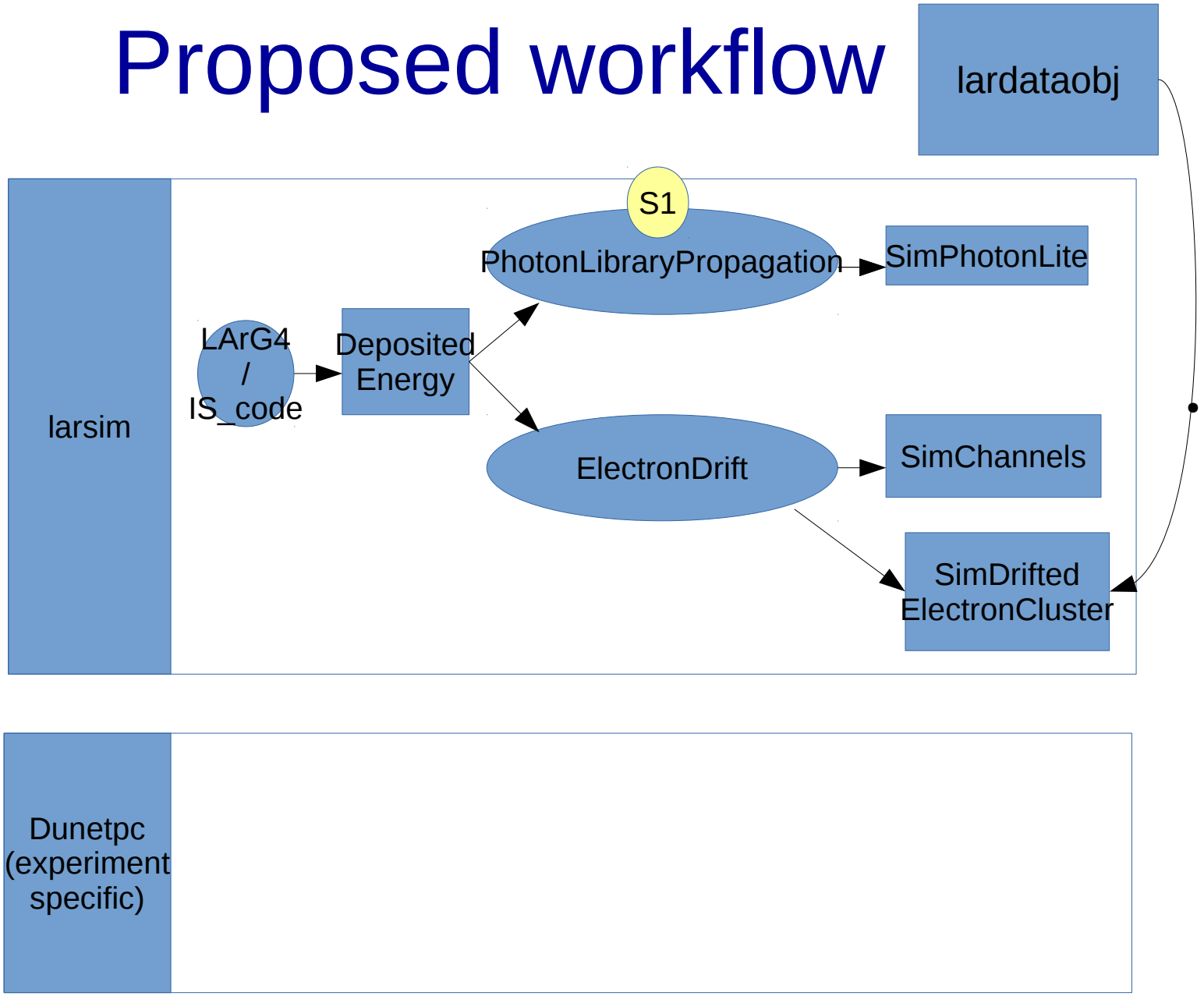
  - Current workflow.
  - Proposed workflow.
  - Changes needed in larsim/larana/lardataobj.
- Bonus: Improvements in S1 time parametrization (extended photon library) (larsim).

Current workflow



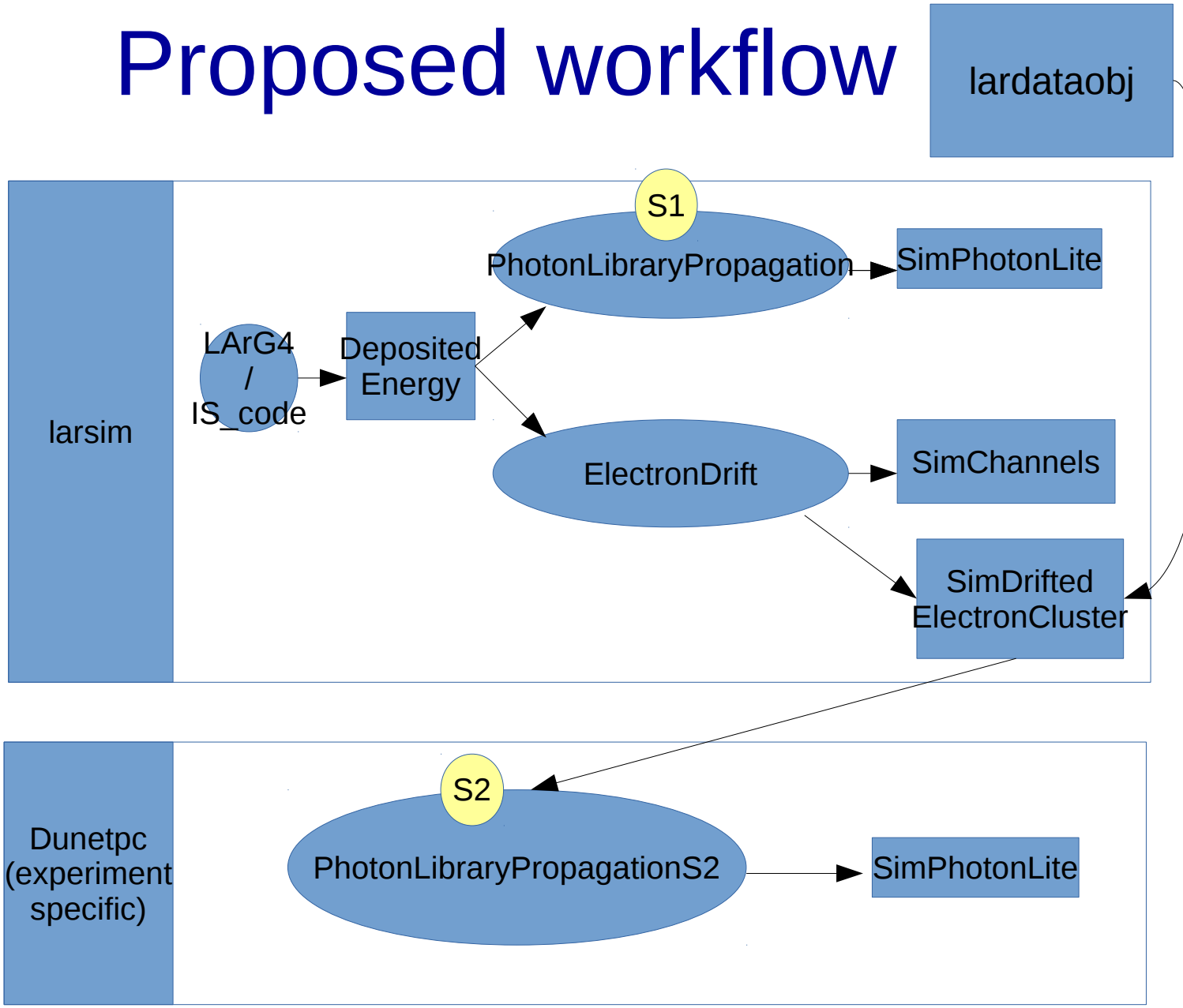
- To simulate the S2 photons we will work over the **FastOptical** workflow. We do not need to have a full simulation solution including S2.
- In the current workflow, photon propagation and electron drift are called in LArG4, but there is plan to rearrange this, having the deposited energy as an intermediate dataproduct: The proposed workflow takes into account this rearragement.

Proposed workflow



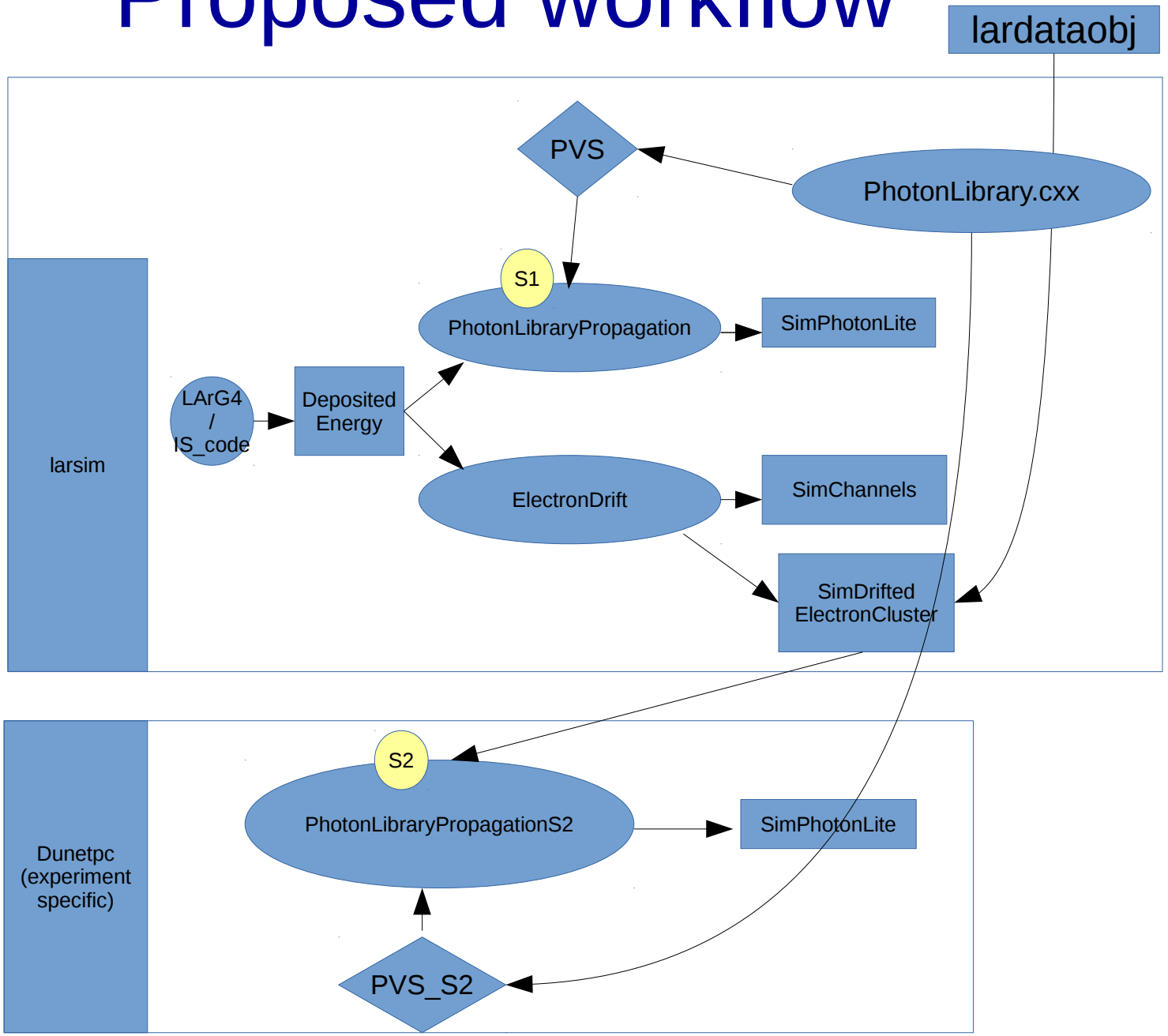
- A **new Data Product** has been created: **DriftedElectronCluster** that contains the information of every drifted electron cluster (size, 3d coordinates, timing).

Proposed workflow



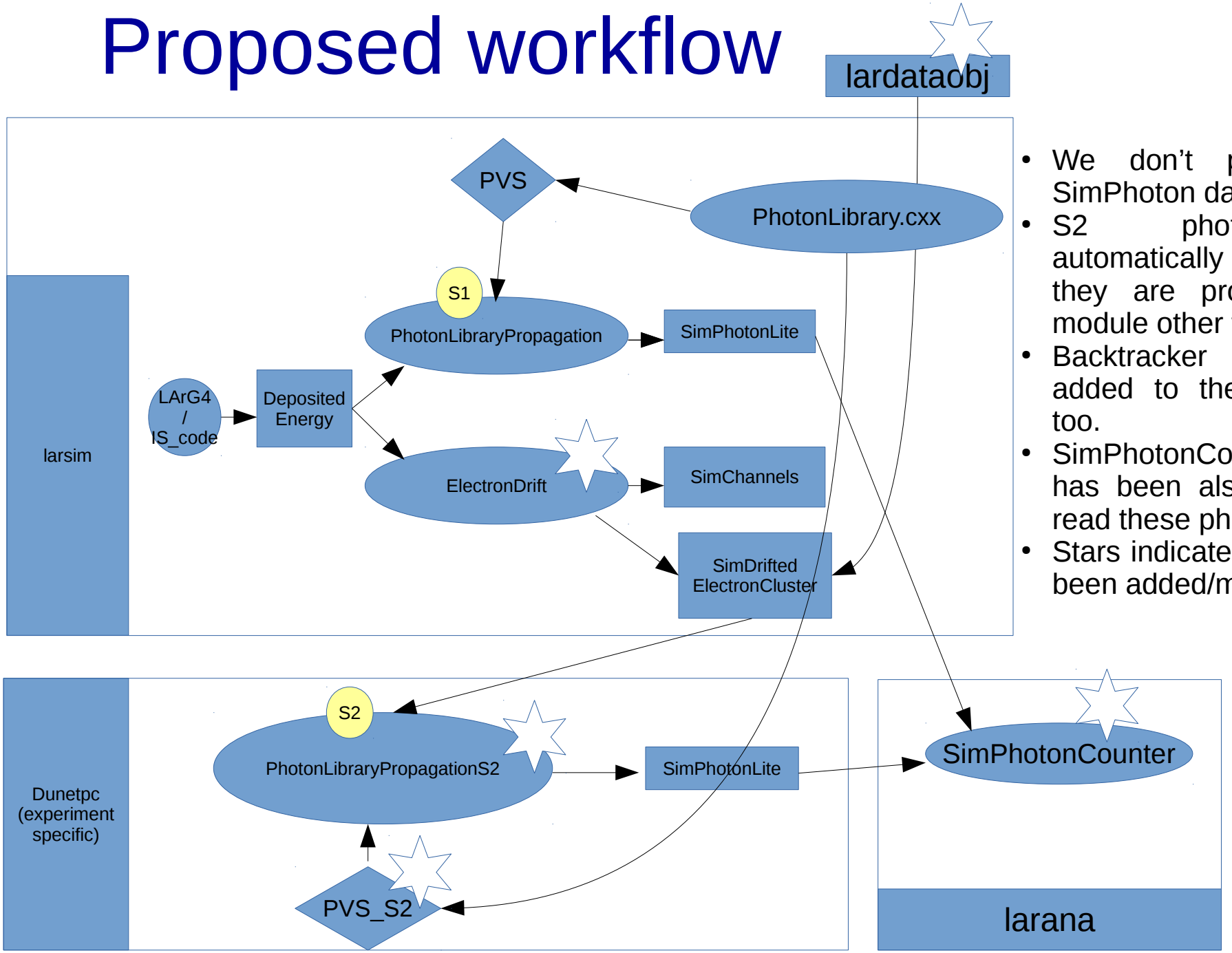
A new **PhotonLibraryPropagationS2** module in dunetpc has been added that reads the ElectronClusters and creates the S2 photons as SimPhotonsLite (the same dataproduct we use to store the S1 photons in Fast Optical simulations).

Proposed workflow



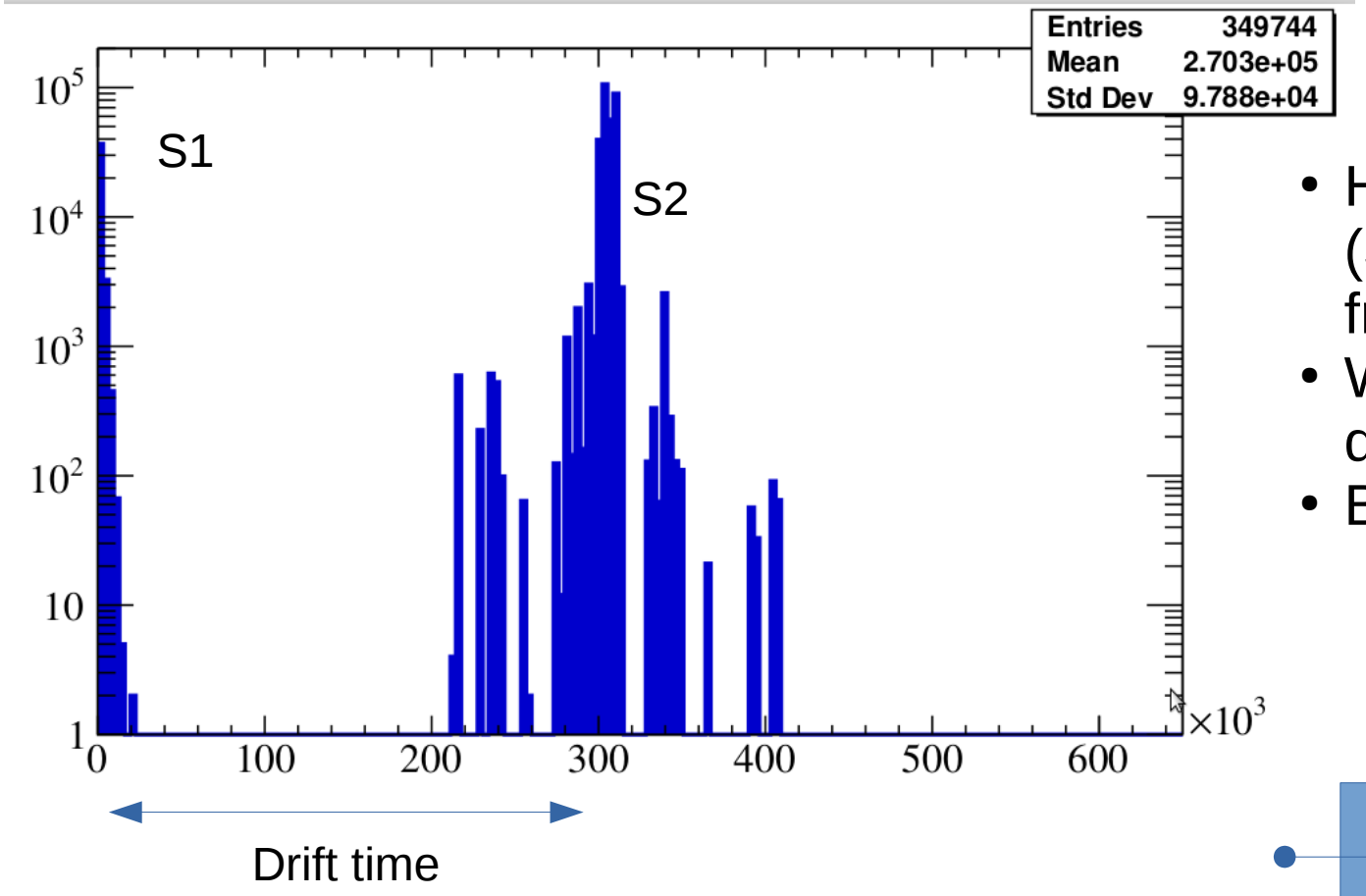
- This PhotonLibrary PropagationS2 module relies on a second photon library. To use it a secondary PhotonVisibilityService has been created in dunetpc.
- S2 photons are created as SimPhotonsLite dataproduct, so they can be directly plugged into the OpDetDigitizer module to generate waveforms and reconstruct, or to SimPhotonCounter.

Proposed workflow

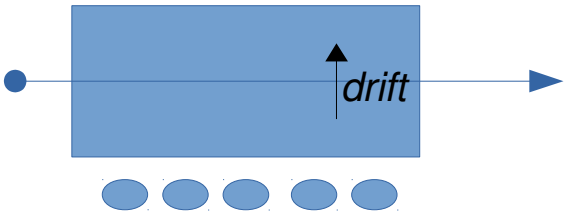


- We don't plan to use SimPhoton data product.
- S2 photons are automatically tagged as they are produced by a module other than *largeant*.
- Backtracker has been added to the S2 photons too.
- SimPhotonCounter module has been also modified to read these photons.
- Stars indicate code that has been added/modified.

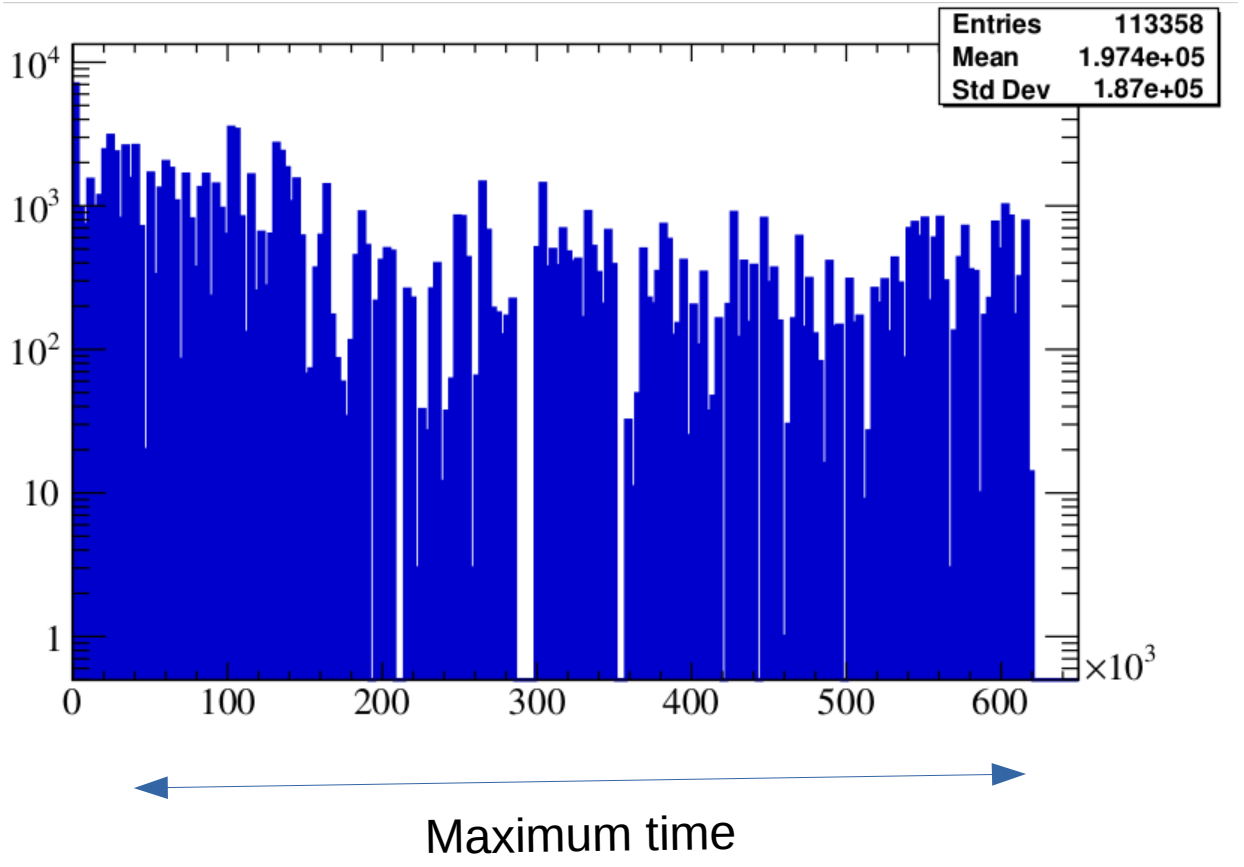
Example in the 3x1x1 geometry



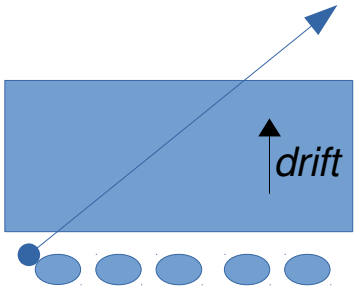
- Horizontal muon (3GeV) at 50cm from the LEMs.
- We expect a 300us drift time.
- Binning of 3.25us.



Example in the 3x1x1 geometry



- Diagonal muon (3GeV) crossing all the detector from the bottom to the top.
- We expect a maximum drift of 1m.
- Binning of 3.25us.



Summary

- **lardataobj repo**: Created a new Data Product: DriftedElectronCluster, that contains the information of every drifted electron cluster (size, 3d coordinates, timing).
- **larsim repo:** The drifted electron cluster information is extracted from larsim/ElectronDrift.
- **dunetpc repo**: Created a new PhotonLibraryPropagationS2 module in dunetpc that creates the S2 photons, and a PhotonVisibilityServiceS2, with an S2 photon library that propagate them towards the photon detectos.
- **larana repo**: Modified SimPhotonCounter, to read these photons.

Modifications in lardataobj
feature/jsoto_SimDriftedElectronCluster

- New dataobject:
  - lardataobj/Simulation/SimDriftedElectronCluster.h

```
namespace sim
{

  // This structure contains all the information per simulated cluster of drifted electrons towards the anode.

  class SimDriftedElectronCluster
  {
  public:
    using Length_t = double;
    using Point_t = ROOT::Math::PositionVector3D< ROOT::Math::Cartesian3D<Length_t> >;

    SimDriftedElectronCluster(int ne = 0,
                              double t = 0.,
                              Point_t start = {0.,0.,0.},
                              Point_t end = {0.,0.,0.},
                              Point_t width = {0.,0.,0.},
                              double e = 0.,
                              int id = 0)
      : NumberOfElectrons(ne)
      , Time(t)
      , InitialPosition(start)
      , FinalPosition(end)
      , ClusterWidth(width)
      , Energy(e)
      , MotherTrackID(id)
    {}
```

Modifications in larsim (S2)
feature/jsoto_dualphase_DriftedElectronClusters

- All changes in Larsim/ElectronDrift/ElectronDrift_module.cc

  - We activate them through a fhicl parameter:

```
fStoreDriftedElectronClusters = p.get< bool >("StoreDriftedElectronClusters", false);
```

  - We define the collection of drifted electrons:

```
if(fStoreDriftedElectronClusters)produces< std::vector<sim::SimDriftedElectronCluster> >();
```

  - We fill it.

```
if(fStoreDriftedElectronClusters)
  SimDriftedElectronClusterCollection->push_back(sim::SimDriftedElectronCluster(
                                                 fnElDiff[k],
                                                 TDiff + simTime,          // timing
                                                 {mp.X(),mp.Y(),mp.Z()}, // mean position of the deposited energy
                                                 {fDriftClusterPos[0],fDriftClusterPos[1],fDriftClusterPos[2]}, // final position of the drifted

                                                 {LDiffSig,TDiffSig,TDiffSig}, // Longitudinal (X) and transverse (Y,Z) diffusion
                                                 fnEnDiff[k], //deposited energy that originated this cluster
                                                 energyDeposit.TrackID()) );
```

  - We store it.

```
if (fStoreDriftedElectronClusters) event.put(std::move(SimDriftedElectronClusterCollection));
```

Modifications in larana
feature/jsoto_photoncounter_generalized

- simphotoncounter_module.cc modified.
- Turning a fhicl parameter into a vector of strings:

  From:

  - fInputModule= pset.get<std::string>("InputModule","largeant");

  To:

  + fInputModule= pset.get<std::vector<std::string>>("InputModule",{"largeant"});

  Then in the code has been turned into a loop:

  - evt.getView(fInputModule, **sccol**); ...

  + for(auto mod : fInputModule){ evt.getView(mod, **sccol**); ... }

- **This is a breaking change! Fhicl files should change their input module including square brackets:**

  InputModule: from "**largeant**" **to [**"**largeant**"**]**

Bonus: Improvements in the extended photon libraries

- **Extended photon libraries where conceived to handle the arrival time information of the photons in the same way it is done with the visibility: The arrival time is fitted with a function for every pair voxel-pmt, and this information is added to the photon library.**
- This solution was presented in May 2018: https://indico.fnal.gov/event/17099/
- Again all changes are under an if statement set to negative by default, that needs to be activated through the timing fhicl parameter. Changes in:
  - PhotonPropagation/PhotonLibrary.cxx/h
  - PhotonPropagation/PhotonVisibilityService_service.cc/h
  - LArG4/OpFastScintillation.cxx
- Changes:
  - Before:
    - The **photon libraries stores the time parameters**, and then the propagation functions are created and called when simulating the photons in LArG4/OpFastScintillation.
    - The time propagation formula is set up by the user through a fhicl parameter.
  - Now:
    - The **photon libraries creates and stores directly the functions** (not the parameters), and then the functions are call every time we need to use them.
    - The **time propagation formula is stored inside the photon library** (to avoid errors), since this is library dependant.
- Results:
  - Faster simulations.
  - Same memory consumption.
  - The same concept will be adapted to S2 once this code has been merged (since the S2 photon visibility service depends on the photon library code).

Bonus: Improvements in the extended photon libraries

Modifications in larsim (S1)

feature/jsoto_ExtendedPhotonLibrary

- New functions and variables:

```
float GetTimingPar(size_t Voxel, size_t OpChannel, size_t parnum) const;
void SetTimingPar(size_t Voxel, size_t OpChannel, float Count, size_t parnum);

///Returns whether the current library deals with time propagation distributions.
int hasTiming() const { return fHasTiming; }
```

```
std::vector<std::vector<float>> fTimingParLookupTable;
```

```
class TF1;

std::vector<std::vector<float>> fTimingParLookupTable;
std::vector<TF1> fTimingParTF1LookupTable;
std::string fTimingParFormula;
size_t fTimingParNParameters;
```

```
/// Unchecked access to a parameter the time distribution
float const& uncheckedAccessTimingPar (size_t Voxel, size_t OpChannel, size_t parnum) const
{ return fTimingParLookupTable[uncheckedIndex(Voxel, OpChannel)][parnum];}

/// Unchecked access to a parameter of the time distribution
float& uncheckedAccessTimingPar(size_t Voxel, size_t OpChannel, size_t parnum)
{ return fTimingParLookupTable[uncheckedIndex(Voxel, OpChannel)][parnum]; }
```

```
/// Unchecked access to a parameter of the time distribution
TF1& uncheckedAccessTimingTF1(size_t Voxel, size_t OpChannel)
{ return fTimingParTF1LookupTable[uncheckedIndex(Voxel, OpChannel)]; }

/// Unchecked access to a parameter of the time distribution
const TF1& uncheckedAccessTimingTF1(size_t Voxel, size_t OpChannel) const
{ return fTimingParTF1LookupTable[uncheckedIndex(Voxel, OpChannel)]; }
```

- Change of LoadLibraryFromFile:

All code is activated with a fhicl parameter.

```
if(fHasTiming)
{
  //for (size_t k=0;k<fTimingParNParameters;k++){ uncheckedAccessTimingPar(Voxel, OpChannel,k) = timing_par[k];}

  TF1 timingfunction(Form("timing_%i_%i",Voxel,OpChannel),fTimingParFormula.c_str(),timing_par[0],100);

  for (size_t k=1;k<fTimingParNParameters;k++)
  {
    timingfunction.SetParameter(k,timing_par[k]);
  }

  uncheckedAccessTimingTF1(Voxel,OpChannel) = timingfunction;
}
} // for entries
```

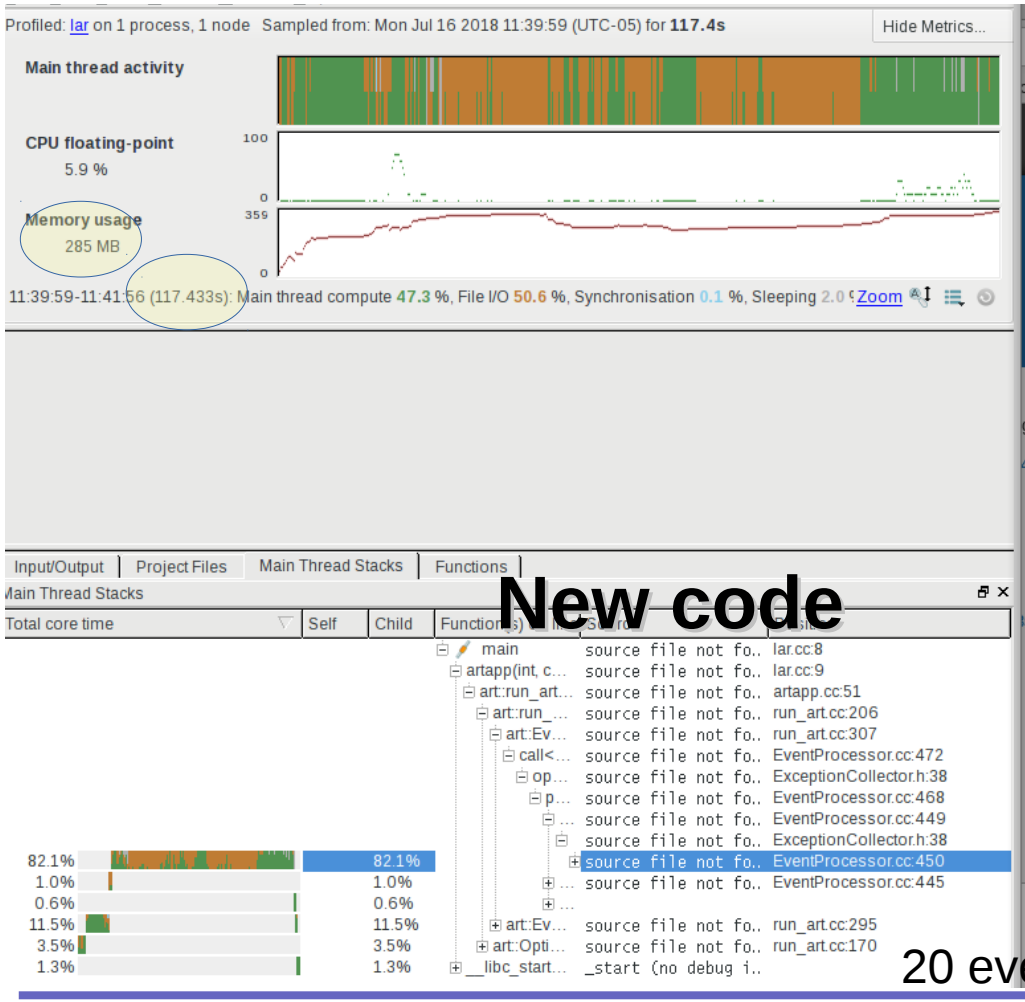
Now we create and initialize the TF1 when we load the library.

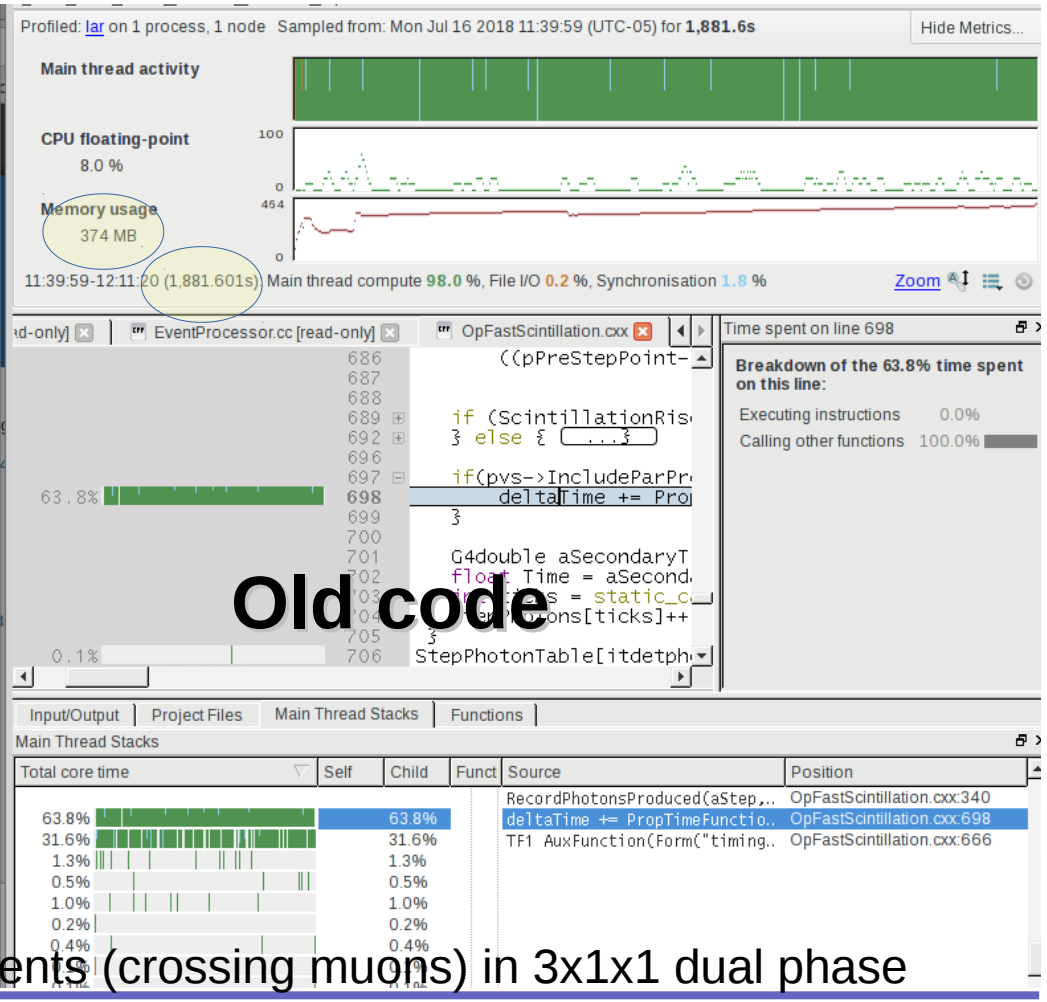
Bonus: Improvements in the extended photon libraries

Modifications in larsim (S1)

feature/jsoto_ExtendedPhotonLibrary

Performance improved: 10 times faster, and same order of memory consumption (even smaller).





20 events (crossing muons) in 3x1x1 dual phase

Modifications in dunetpc

- New PhotonLibraryPropagationS2 module added.
- New PhotonVisibilityServiceS2 added (duplicated), as we cannot use it in the same run with different fhicl parameters.

Next steps

- I propose to merge these four branches, and continue working in the S2 simulation within the dunetpc repository.
  - **feature/jsoto_SimDriftedElectronCluster in lardataobj.**
  - **feature/jsoto_dualphase_DriftedElectronClusters in larsim.**
  - **feature/jsoto_photoncounter_generalized in larana**
  - **feature/jsoto_ExtendedPhotonLibrary in larsim**
- We will also hook up this with the new artg4tk+larg4 workflow (we have already started to talk with Hans).
- Converge the S1 and S2 photon visibility service: It might be a tool?
- Studies of the scalability to the far detector.

Thanks a lot to Paul!