

# Development of CEPC Tracking Algorithms

SUN Shengsen

On Behalf of CEPC Software Working Group

IAS Program on High Energy Physics 2020 Conference

Hong Kong, January 20-22

# Introduction of CEPC Software Prototype

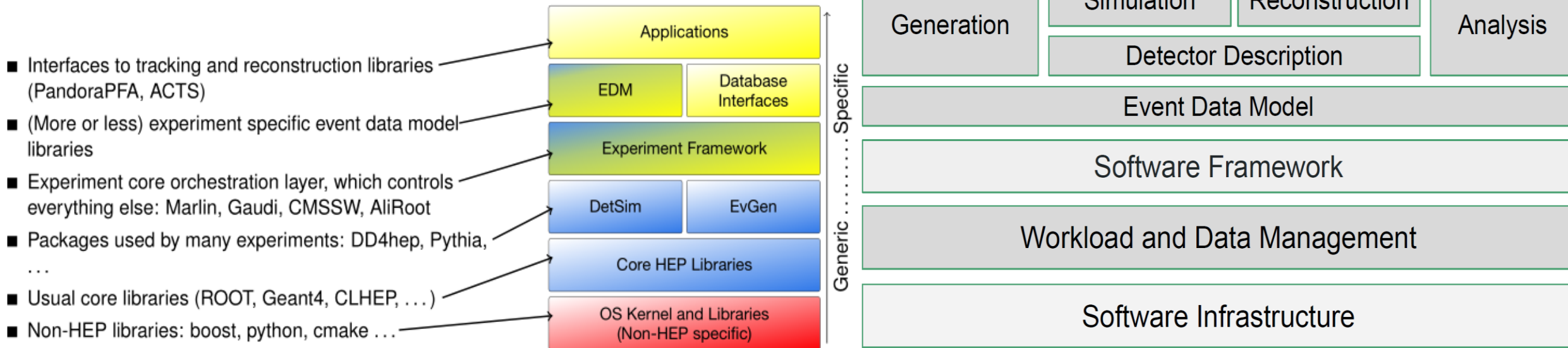
- CEPC software originally started from the iLCSoft (many thanks)
  - LCIO, Marlin, tracking and flavor-tagging
  - New components for CEPC: simulation, reconstruction...
  - Used for the CDR study, which is released in Nov, 2018
- A new framework for TDR is considered at the CEPC Oxford workshop, April 2019
  - To demonstrate the capability to meet the future requirements
  - To support continuous integrations of new software components
- Reached the consensus at the Bologna workshop, June 2019
  - Investigate the possibility to have a common Event Data Model (EDM4Hep)
  - Contribute to the development of a Common Turnkey Software Stack (Key4hep)
    - ILC, CLIC, FCC, CEPC, SCTF
  - Maximize the sharing of software components between experiments

# A typical HEP Software Stack

[Ref]: André Sailer, etc. , CHEP2019

[https://indico.cern.ch/event/773049/contributions/3474763/attachments/1938664/3213633/191105\\_sailer\\_key4hep.pdf](https://indico.cern.ch/event/773049/contributions/3474763/attachments/1938664/3213633/191105_sailer_key4hep.pdf)

Applications usually rely on large number of libraries, where some depend on others



[Ref]: Gerri GANIS, Mini-Workshop: Experiment / Detector-Software and Physics Requirements for  $e^+e^-$  Colliders

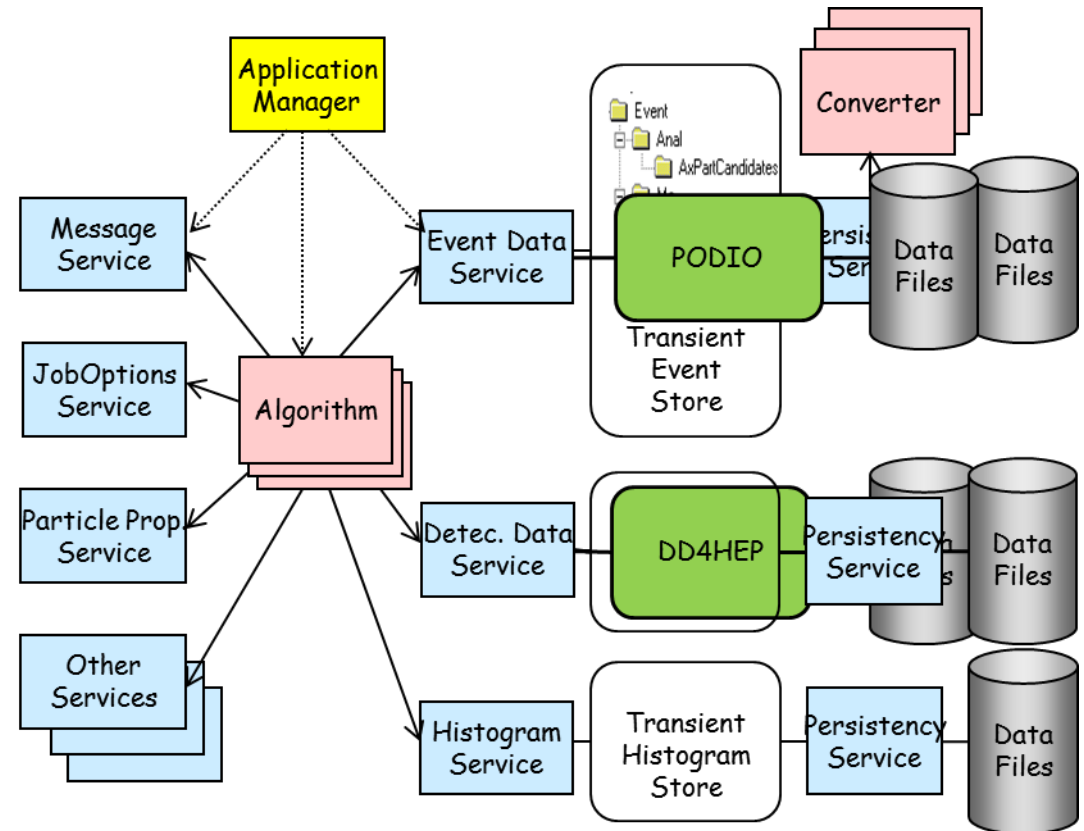
[http://ias.ust.hk/program/shared\\_doc/2020/202001hep/workshop/exp/20200117\\_1038\\_am\\_Gerri\\_GANIS.pdf](http://ias.ust.hk/program/shared_doc/2020/202001hep/workshop/exp/20200117_1038_am_Gerri_GANIS.pdf)

# Status of CEPCSW Prototype

- Goal: Based on **Key4HEP** (Common Turnkey Software Stack)
- Provide a **ready-to-work environment** to algorithm developers and physicists.
- CEPCSW prototype is developed based on **Gaudi**
- **PLCIO** as **the transitional EDM** from LCIO to EDM4HEP.
  - Extension to read **LCIO data** generated by Marlin
- **DD4Hep** for detector simulation
- Porting of available algorithms from Marlin to Gaudi
- Both detector simulation and tracking algorithm can run successfully

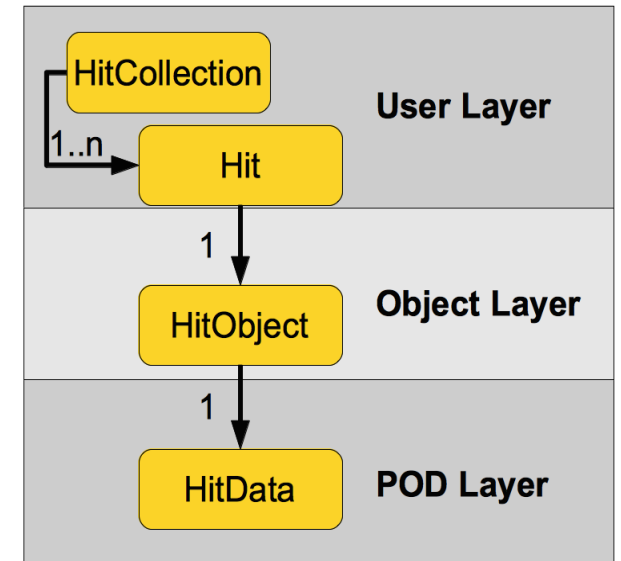
# Gaudi: the Underlying Framework

- The core part of the framework is small
- Key components:
  - Application manager
  - Services
  - Algorithms
  - Tools
- Data is separated from algorithms – physicists could concentrate on the algorithms
- High performance computing
  - Multithreading computing is supported since v 29
  - Parallelized functional and reentrant algorithms
  - Transparent data management in memory



# PODIO

- PODIO is an EDM toolkit developed in AIDA2020 EU-project
- Based on the idea of using PODs for the event data objects (Plain-Old-Data object)
  - Gain simplicity and performance for memory and I/O operations
- PODIO originally developed in context of the FCC study
- Three layers
  - User layer (API)
  - Object layer
  - POD layer
- Clear design of ownership
- Allow to have 1-1, 1-N, or N-M relationships
- Code generation (C++/Python) for EDM class from yaml files



[Ref]:Frank Gaede, Mini-Workshop: Experiment / Detector-Software and Physics Requirements for e<sup>+</sup>e<sup>-</sup> Colliders

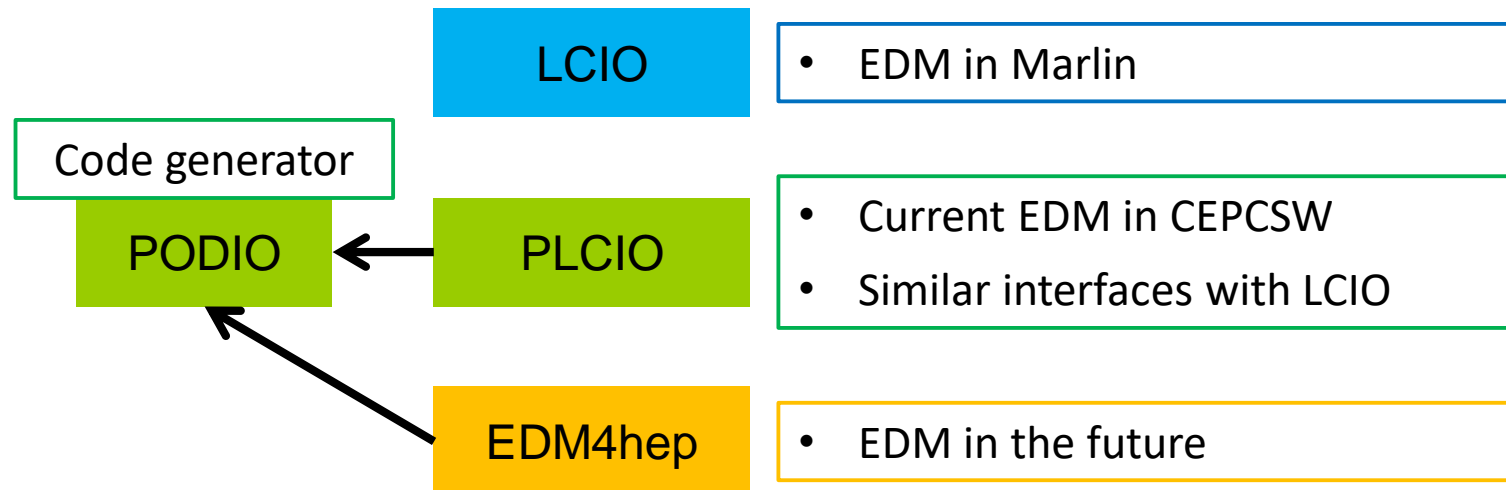
[http://ias.ust.hk/program/shared\\_doc/2020/202001hep/workshop/exp/20200117\\_1038\\_am\\_Frank\\_GAEDE.pdf](http://ias.ust.hk/program/shared_doc/2020/202001hep/workshop/exp/20200117_1038_am_Frank_GAEDE.pdf)

# EDM4HEP

- Common event data model for future (lepton) collider studies will provide foundation of the Turnkey Software Stack
  - Common core classes described in a yaml file
  - C++ code is generated by PODIO
  - The persistency layer (ROOT, HDF5, ...) can be changed easily
  - Each experiment can implement their own extensions
- Use PODIO EDM toolkit and rely on experience with LCIO and FCC-dem
- Implemented a simulation EDM: *MCParticle*, *SimTrackerHit* and *SimCalorimeterHit*
  - Reconstruction part to follow soon
- Carried out as a HEP software foundation project
- A substitute is necessary for CEPCSW at present

# Current EDM in CEPCSW Prototype

- PLCIO is the transitional EDM from LCIO to EDM4HEP in the prototype
- PLCIO is an implement complete LCIO EDM (almost) in PODIO

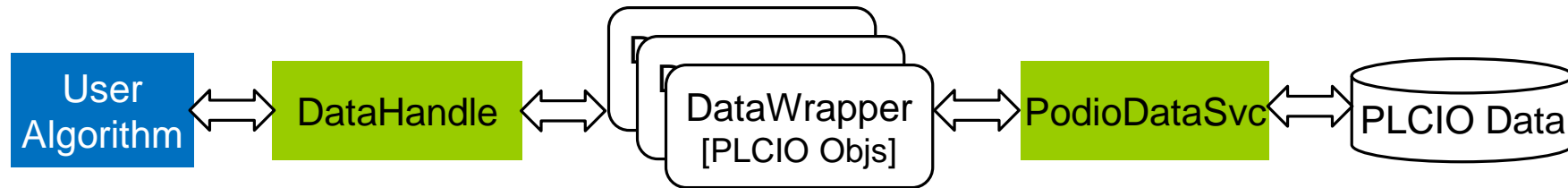


- CEPCSW is the first user of PLCIO
  - A few classes in LCIO are not present in PLCIO now
  - ObjectID in PLCIO is not straightforward to retrieve the correlated object
  - Going to develop helper class or Gaudi service to facilitate event navigation with data objects' relations

# FWCore

- FCCSW FWCore

- DataWrapper FCC-dem data collection → DataOjection in Gaudi
- DataHandle: user interface to register/retrieve data to/from Gaudi TES (Transient Event Store)
- PODIO data service: read/write PODIO data objects

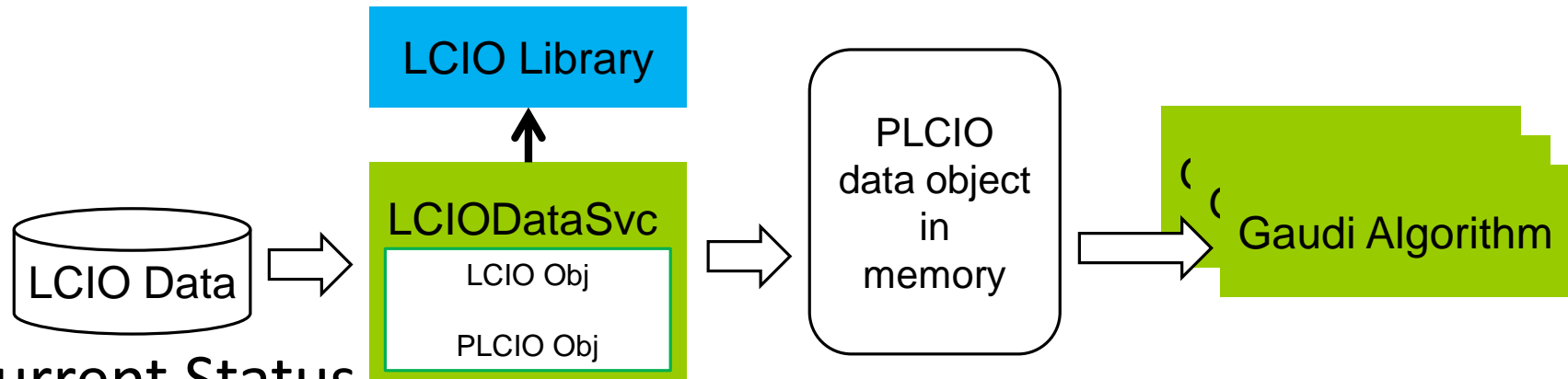


- CEPCSW FWCore

- Mainly taken from FCCSW FWCore (Many thanks)
- Extension to read LCIO data generated by Marlin

# Read the Existing LCIO Data

- LCIODataSvc
  - Read LCIO files via the LCIO library
  - Convert LCIO data objects to PLCIO data objects
  - Register PLCIO data objects to Gaudi Event Data Store

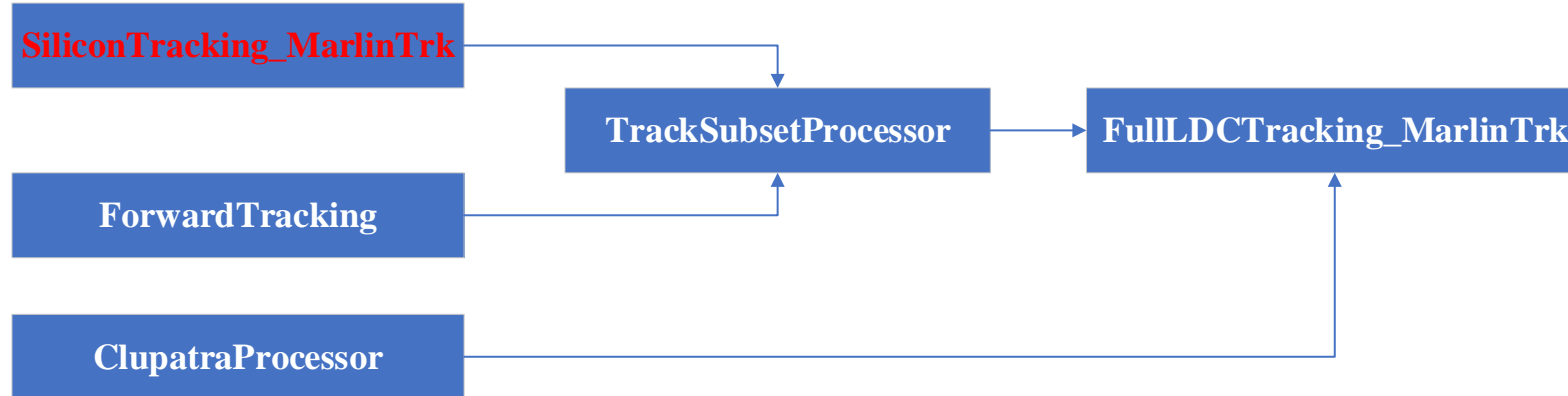


- Current Status
  - Most LCIO data types can be retrieved as PLCIO objects in CEPCSW
  - Some of the data relations are not fully recovered (there are some limitation for data analysis now)



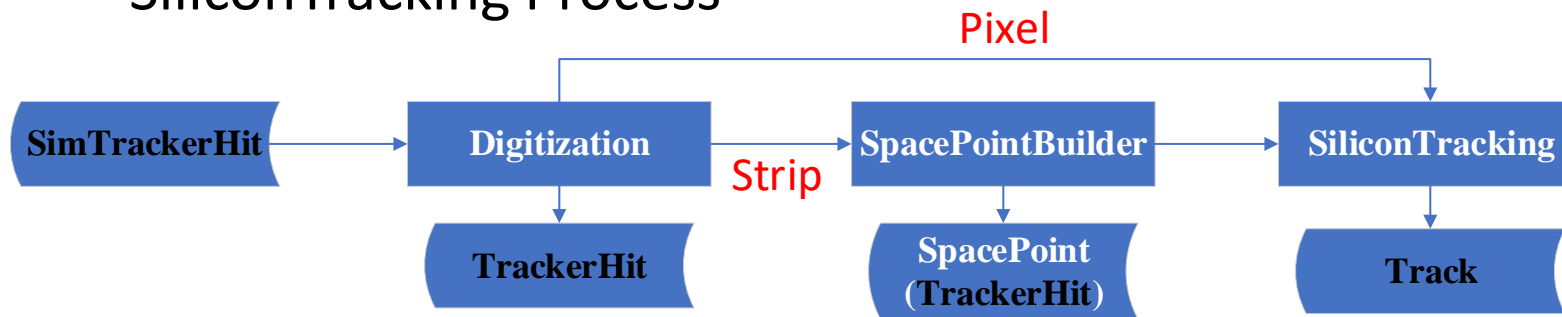
# Porting Tracking Algorithms from Marlin to Gaudi

- Tracking Processes in CDR (Marlin, many thanks)

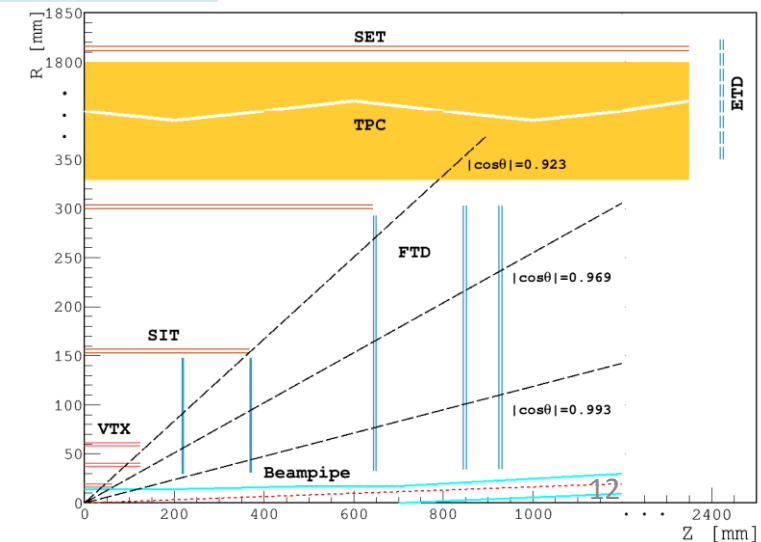


SiliconTracking\_MarlinTrk is chosen as the first porting reconstruction, since it has less dependency and the tracking for silicon detector is more simple than TPC.

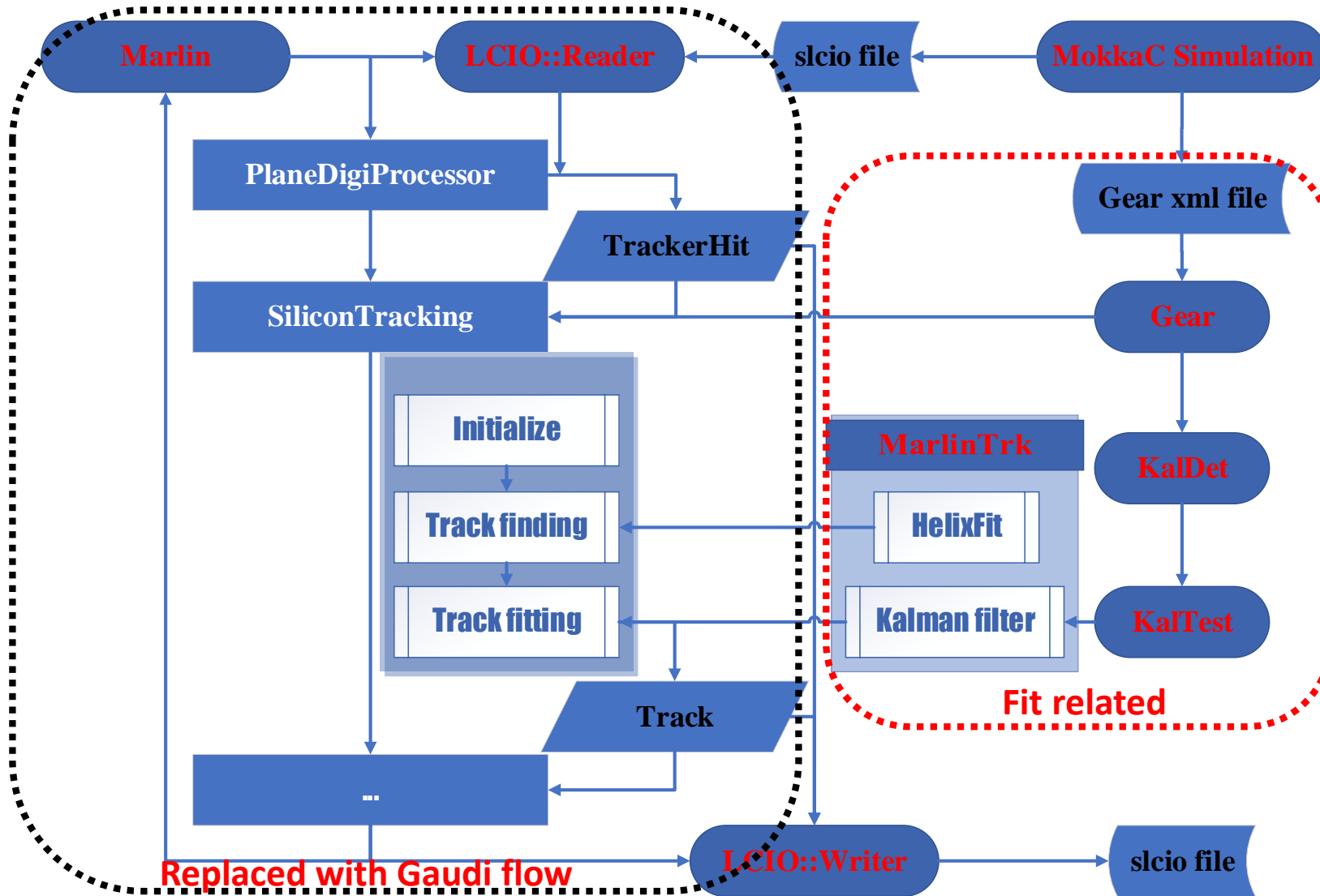
- SiliconTracking Process



SiliconTracking for vertex detector (pixel VXD) only (without strip SIT) is most simple option.



# Working Flow



# Fit Related Packages: Service

- Gear (external library)
  - Loaded by a Gaudi service (GearSvc)


```
class IGearSvc: virtual public IService {
public:
  DeclareInterfaceID(IGearSvc, 0, 1); // major/minor version

  virtual ~IGearSvc() = default;

  // Get the GEAR Manager
  virtual gear::GearMgr* getGearMgr() = 0;

};
```

In Marlin:  
static gear::GearMgr\* GEAR;



- MarlinTrk

- Interface to fitter, loaded by a Gaudi service (TrackSystemSvc)

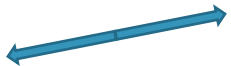
```
class ITrackSystemSvc: virtual public IService {
public:
  DeclareInterfaceID(ITrackSystemSvc, 0, 1); // major/minor version

  virtual ~ITrackSystemSvc() = default;

  //Get the track manager
  virtual MarlinTrk::IMarlinTrkSystem* getTrackSystem() = 0;

  virtual void removeTrackSystem() = 0;
};
```

In Marlin:  
class Factroy{  
public:  
 static IMarlinTrkSystem\* createMarlinTrkSystem(const std::string& systemType,  
 const gear::GearMgr\* mgr ,  
 const std::string& options );



# Fit Related Packages: Library

- MarlinTrk
  - HelixFit
  - HelixTrack
  - MarlinKalTest (IMarlinTrkSystem)
  - MarlinKalTestTrack (IMarlinTrack)
  - MarlinTrkUtils  $\Rightarrow$  non-class function: to call Kalman filter fit (KalTest)
    - createFinalisedLCIOTrack
    - createPrefit
    - createFit
    - finaliseLCIOTrack
    - addHitNumbersToTrack
  - ~~DiagnosticsController~~
  - ~~LCIOTrackPropagators~~
  - ~~MarlinTrkDiagnostics~~
- Dependencies
  - LCIO  $\Rightarrow$  plcio
  - GEAR
  - ROOT
  - CLHEP
  - KalTest
  - KalDet
  - streamlog
- CMakeLists.txt

such as:

```
gaudi_add_library(TrackSystemSvcLib ${TrackSystemSvcLib_srcs}
PUBLIC_HEADERS TrackSystemSvc
INCLUDE_DIRS GaudiKernel ROOT CLHEP gear ${LCIO_INCLUDE_DIRS}
LINK_LIBRARIES GaudiKernel ROOT CLHEP $ENV{GEAR}/lib/libgear
)
```

```
class MarlinKalTestTrack : public MarlinTrk::IMarlinTrack {
public:
    MarlinKalTestTrack(MarlinKalTest* ktest) ;
    ~MarlinKalTestTrack() ;

private:
    MarlinKalTestTrack(const MarlinKalTestTrack&) ;
    MarlinKalTestTrack& operator=(const MarlinKalTestTrack&);

    int addHit(plcio::TrackerHit* hit) ;
    int addHit(plcio::TrackerHit* trkhit, const ILDVMeasLayer* ml) ;
    ...
}
```



```
int addHit(EVENT::TrackerHit* hit) ;
int addHit(EVENT::TrackerHit* trkhit, const ILDVMeasLayer* ml) ;
```

# Fit Related Packages: External

- **KalTest**: provides base Kalman filter support, dependent on ROOT only

- geomlib/
- kallib/
- kaltracklib/
- utils/

re-compile and link in CEPCSW environment

- **KalDet**: geometry support package through GEAR

- gen/
- ild/
- kern/
- lctpc/
- othertpc/

change data model dependency from LCIO to plcio

```
class ILDVMeasLayer : public TVMeasLayer {  
    ...  
    /** Convert LCIO Tracker Hit to an ILDPlanarTrackHit */  
    virtual ILDVTrackHit* ConvertLCIOTrkHit(plcio::TrackerHit* trkhit) const=0;  
    ...  
    virtual ILDVTrackHit* ConvertLCIOTrkHit( EVENT::TrackerHit* trkhit) const = 0 ;
```

- Migrated as external packages
- **streamlog**: library in ILCUTIL package
  - does not depend on any external package, therefore does not need re-compile
  - Will be updated to CEPCSW Gaudi print interface

# From SiliconTracking\_Marlin to SiliconTrackingAlg

- Since **MarlinTrk**, **KalDet** and **KalTest** are all migrated and kept in CEPCSW at first step, less modification needed
- Main algorithm **SiliconTrackingAlg**

```
class SiliconTracking : public GaudiAlgorithm {  
    friend class AlgFactory<SiliconTracking>;  
public:  
    SiliconTracking(const std::string& name, ISvcLocator* svcLoc);  
    virtual StatusCode initialize() ;  
    virtual StatusCode execute() ;  
    virtual StatusCode finalize() ;
```

protected:

...

```
MarlinTrk::HelixFit* _fastfitter;
```

```
gear::GearMgr* _GEAR;
```

```
MarlinTrk::IMarlinTrkSystem* _trksystem ;
```

```
Gaudi::Property<bool> _MSOn{this, "MultipleScatteringOn", true};
```

```
Gaudi::Property<bool> _ElossOn{this, "EnergyLossOn", true};
```

```
Gaudi::Property<bool> _SmoothOn{this, "SmoothOn", true};
```

...

```
DataHandle<plcio::EventHeaderCollection> _headerCol {"EventHeaderCol", Gaudi::DataHandle::Reader, this};
```

```
DataHandle<plcio::TrackerHitCollection> _inVTXColHdl {"VXDTrackerHits", Gaudi::DataHandle::Reader, this};
```

```
DataHandle<plcio::TrackerHitCollection> _inFTDPixelColHdl {"FTDPixelTrackerHits", Gaudi::DataHandle::Reader, this};
```

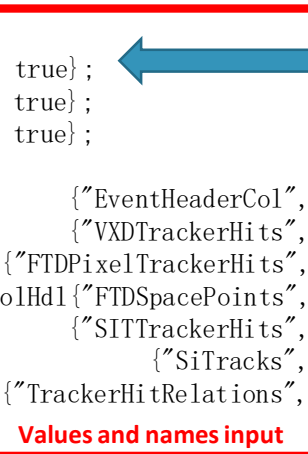
```
DataHandle<plcio::TrackerHitCollection> _inFTDSpacePointColHdl {"FTDSpacePoints", Gaudi::DataHandle::Reader, this};
```

```
DataHandle<plcio::TrackerHitCollection> _inSITColHdl {"SITTrackerHits", Gaudi::DataHandle::Reader, this};
```

```
DataHandle<plcio::TrackCollection> _outColHdl {"SiTracks", Gaudi::DataHandle::Writer, this};
```

```
DataHandle<plcio::LCRelationCollection> _outRelColHdl {"TrackerHitRelations", Gaudi::DataHandle::Writer, this};
```

...



parameters input

data input and output

Values and names input

- Support classes (**CaloHitExtended ClusterExtended GroupTracks HelixClass LineClass TrackerHitExtended TrackExtended**) from **MarlinUtil**
  - Change data model from LCIO to plcio, just like **MarlinTrk** and **KalDet**

# Difference of Usage with Marlin Processor

```
SiliconTracking_MarlinTrk::SiliconTracking_MarlinTrk() : Processor("SiliconTracking_MarlinTrk") {  
    ...  
    registerProcessorParameter("MultipleScatteringOn", "Use MultipleScattering in Fit", _MSOn, bool(true));  
    ...  
}  
  
void SiliconTracking_MarlinTrk::init() {  
    ...  
    _trksystem = MarlinTrk::Factory::createMarlinTrkSystem( "KalTest", marlin::Global::GEAR, "" );  
    if( _trksystem == 0 ) {  
        throw EVENT::Exception( std::string("Cannot initialize MarlinTrkSystem of Type: KalTest" ) );  
    }  
    _trksystem->setOption( IMarlinTrkSystem::CFG::useQMS, _MSOn );  
    _trksystem->setOption( IMarlinTrkSystem::CFG::usedEdx, _ElossOn );  
    _trksystem->setOption( IMarlinTrkSystem::CFG::useSmoothing, _SmoothOn );  
    _trksystem->init();  
    ...  
}  
  
void SiliconTracking_MarlinTrk::processEvent( LCEvent * evt ) {  
    ...  
    try {  
        LCCollection * hitCollection = evt->getCollection("VXDTrackerHits");  
        int nelelem = hitCollection->getNumberOfElements();  
        for (int ielem=0; ielem<nelelem; ++ielem) {  
            TrackerHitPlane* hit = dynamic_cast<EVENT::TrackerHitPlane*>(hitCollection->getElementAt(ielem));  
            ...  
        }  
    }  
    catch(DataNotAvailableException &e){  
        ...  
    }  
    ...  
    bool backwards = IMarlinTrack::backward;  
    MarlinTrk::IMarlinTrack* marlinTrk = _trksystem->createTrack();  
    int error = 0;  
    try {  
        error = MarlinTrk::createFinalisedLCIOTrack(marlinTrk, trkHits, track, backwards, cov, _bField, _maxChi2);  
    }  
    catch (...){  
        ...  
    }  
    ...  
}
```

```
SiliconTracking::SiliconTracking(const std::string& name, ISvcLocator* svcLoc) : GaudiAlgorithm(name, svcLoc) {  
    ...  
    declareProperty("HeaderCol", _headerCol);  
    declareProperty("VTXHitCollection", _inVTXColHdl, "Handle of the Input VTX TrackerHits collection");  
    ...  
}  
  
StatusCode SiliconTracking::initialize() {  
    ...  
    auto _trackSystemSvc = service<ITrackSystemSvc>("TrackSystemSvc");  
    if ( !_trackSystemSvc ) {  
        error() << "Failed to find TrackSystemSvc ..." << endmsg;  
        return StatusCode::FAILURE;  
    }  
    _trksystem = _trackSystemSvc->getTrackSystem();  
    if( _trksystem == 0 ) {  
        error() << "Cannot initialize MarlinTrkSystem of Type: KalTest" <<endmsg;  
        return StatusCode::FAILURE;  
    }  
    _trksystem->setOption( IMarlinTrkSystem::CFG::useQMS, _MSOn );  
    _trksystem->setOption( IMarlinTrkSystem::CFG::usedEdx, _ElossOn );  
    _trksystem->setOption( IMarlinTrkSystem::CFG::useSmoothing, _SmoothOn );  
    _trksystem->init();  
    ...  
}  
  
StatusCode SiliconTracking::execute(){  
    ...  
    const plcio::TrackerHitCollection* hitVTXCol = nullptr;  
    try {  
        hitVTXCol = _inVTXColHdl.get();  
    }  
    catch ( GaudiException &e ) {  
        debug() << "Collection " << _inVTXColHdl.fullKey() << " is unavailable in event " << _nEvt << endmsg;  
        success = 0;  
    }  
    if(hitVTXCol){  
        int nelelem = hitVTXCol->size();  
        for (int ielem=0; ielem<nelelem; ++ielem) {  
            plcio::TrackerHit hit = hitVTXCol->at(ielem);  
            ...  
        }  
    }  
    ...  
    bool fit_backwards = IMarlinTrack::backward;  
    MarlinTrk::IMarlinTrack* marlinTrk = _trksystem->createTrack();  
    int status = 0;  
    try {  
        status = MarlinTrk::createFinalisedLCIOTrack(marlinTrk, trkHits, &track, fit_backwards, covMatrix, _bField, _maxChi2PerHit);  
    } catch (...) {...}  
    ...  
}
```

Interface load

# Problems and Temporary Solutions

- Can't get relative object directly in **plcio** data model
  - Remove relative object usage in tracking and after fitting, influence some following analysis based on **MC truth** particles
  - Going to extend plcio with a service to handle the object relations
- Can't convert pointers of data objects each other
  - Fix data type in algorithm and use template in codes of module classes
  - Only use one type tracker hit as temporary now, saving U,V directions and resolutions into CovMatrix

```
auto trkHit = trkhitVec->create();
std::array<float, 6> cov;
cov[0] = u_direction[0];
cov[1] = u_direction[1];
cov[2] = resU;
cov[3] = v_direction[0];
cov[4] = v_direction[1];
cov[5] = resV;
trkHit->setCovMatrix(cov);
```



```
EVENT::TrackerHitPlaneImpl* trkHit = new EVENT::TrackerHitPlaneImpl ;
trkHit->setU( u_direction ) ;
trkHit->setV( v_direction ) ;
trkHit->setdU( resU ) ;
trkHit->setdV( resV ) ;
```

- From TrackerHit pointer (LCIO) to TrackerHit object (plcio)
  - Plcio's hit object does not have data member, just only data object pointer
  - In order to keep pointer usage in KalDet and TrackSystemSvc, build temporary object vector for each event
    - `_allHits.reserve(100000);`
    - `_allHits.push_back(hit); // &hit can be used as plcio::TrackerHit* before clear()`
    - `_allHits.clear();`
  - Will be updated to object transmission
- Geometry service during the transition from Marlin to Gaudi
  - Keep the GEAR package in Marlin at present, similar to load as TrackSystemSvc

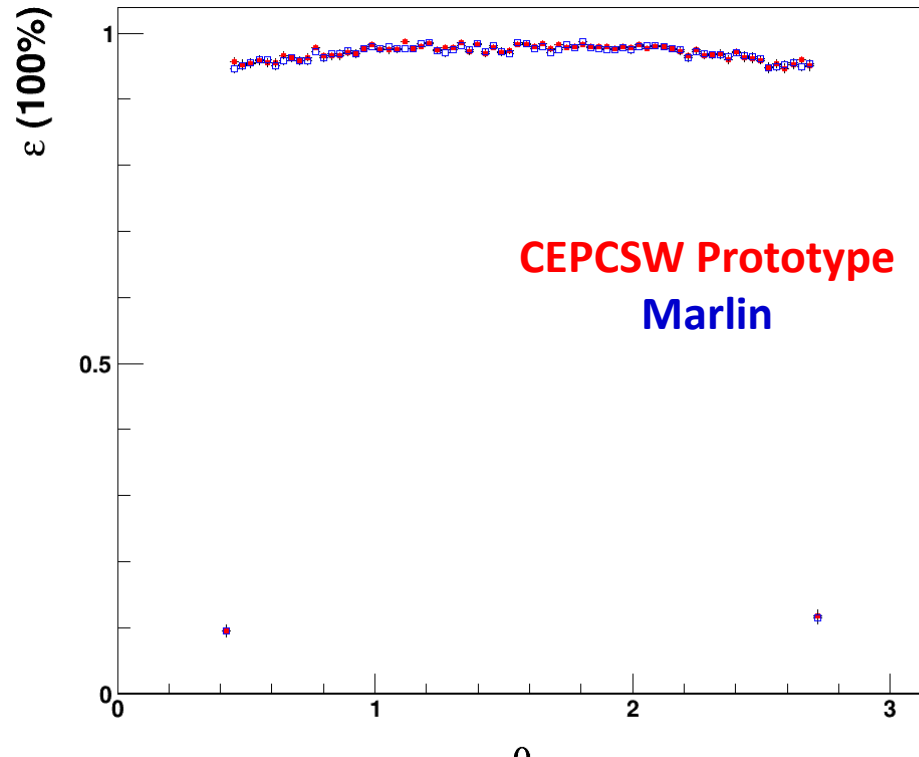
# Tracking Efficiency

- Definition

- $\epsilon = N_{\text{matched\_track}} / N_{\text{MC(primary)}}$

- Matching:

- $|\text{par}_{\text{fit}} - \text{par}_{\text{MC}}| < 5\sigma_{\text{par}}$  (par=d0, phi0,  $\omega$ , z0, tan $\lambda$ )



	$R$ (mm)	$ z $ (mm)	$ \cos\theta $	$\sigma$ ( $\mu\text{m}$ )
Layer 1	16	62.5	0.97	2.8
Layer 2	18	62.5	0.96	6
Layer 3	37	125.0	0.96	4
Layer 4	39	125.0	0.95	4
Layer 5	58	125.0	0.91	4
Layer 6	60	125.0	0.90	4

**Fake rate:**

CEPCSW:  $(1.23 \pm 0.04)\%$

Marlin:  $(1.21 \pm 0.04)\%$

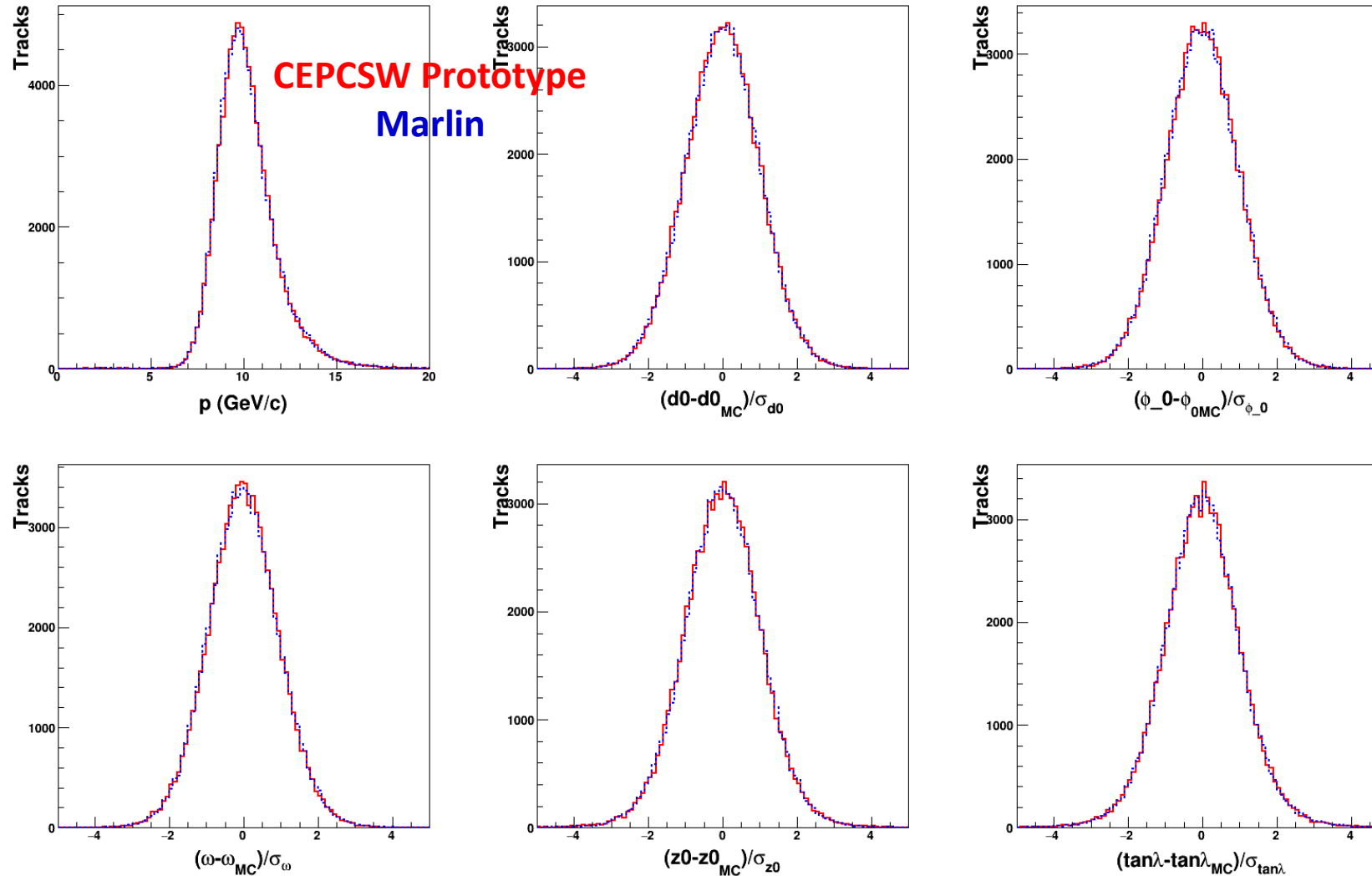


10 $\sigma$  cut:

CEPCSW:  $(0.56 \pm 0.03)\%$

Marlin:  $(0.56 \pm 0.03)\%$

# Resolution and Pull



CEPCSW's results are consistent with Marlin's, but they are not identical.

# Comparison of Tracking and Fitting

- Single muon particle ( $p=10$  GeV/c,  $\theta \in [10^\circ, 170^\circ]$ ,  $\phi \in [0^\circ, 360^\circ]$ )
  - same input (simulated by MokkaC with only VXD)
  - same reconstruction options
- The difference is caused by different random number in digitization (gaussian smear)
  - if testing by set **space resolution as 0**, and debugging, they are fully identical
  - CEPCSW

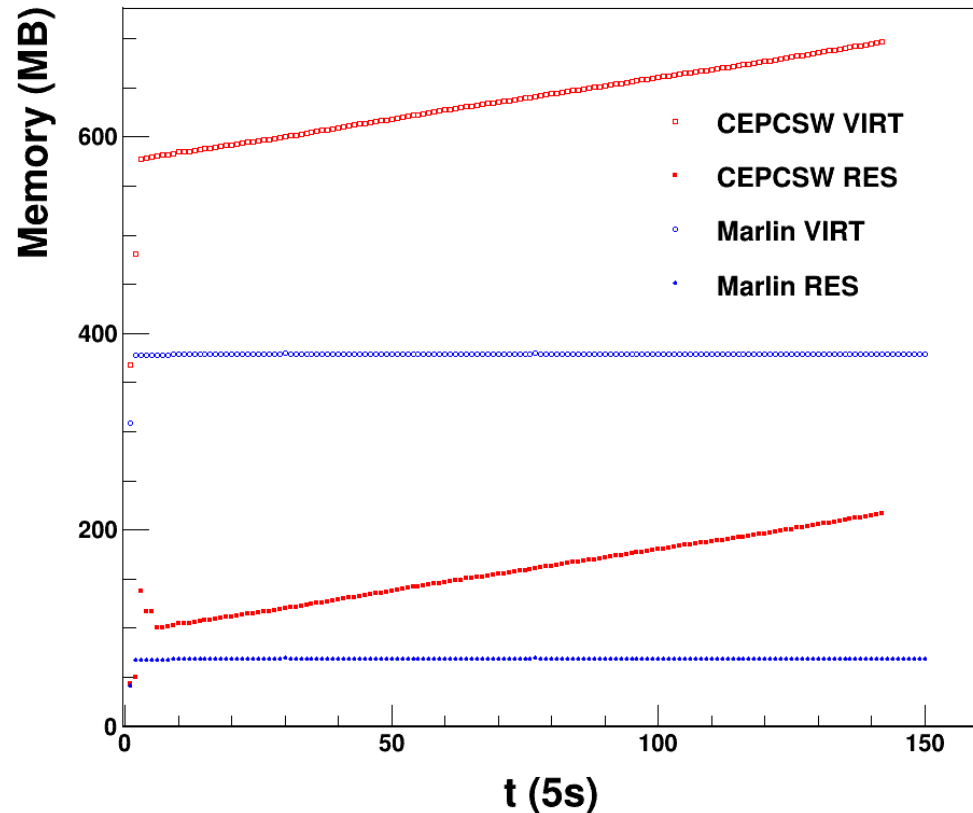
```
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 375.662 Px = -0.703028 Py = -375.661 Pz = -0.0104615
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 453.497 Px = 196.305 Py = 408.808 Pz = -0.0370978
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 1205.55 Px = 661.275 Py = -1008 Pz = 0.128028
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 488.419 Px = 486.218 Py = 46.3153 Pz = -0.0540399
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 2622.19 Px = -1277.83 Py = -2289.76 Pz = -0.177663
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 2642.46 Px = 2284.13 Py = 1328.66 Pz = 0.146478
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 656.724 Px = 18.8581 Py = -656.453 Pz = -0.0383404
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 512.21 Px = 119.476 Py = -498.081 Pz = -0.0241696
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 2563.15 Px = -1801.14 Py = -1823.63 Pz = -0.0922066
SiliconTracking  DEBUG Total 4-momentum of Si Tracks : E = 405.058 Px = -380.559 Py = -138.733 Pz = 0.018916
```

- Marlin

```
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 3.756618e+02 Px = -7.335976e-01 Py = -3.756611e+02 Pz = -1.046148e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 4.534971e+02 Px = 1.963345e+02 Py = 4.087937e+02 Pz = -3.709782e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 1.205546e+03 Px = 6.612491e+02 Py = -1.008014e+03 Pz = 1.280278e-01
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 4.884193e+02 Px = 4.862154e+02 Py = 4.634652e+01 Pz = -5.403987e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 2.622189e+03 Px = -1.277834e+03 Py = -2.289763e+03 Pz = -1.776625e-01
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 2.642462e+03 Px = 2.284150e+03 Py = 1.328632e+03 Pz = 1.464776e-01
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 6.567236e+02 Px = 1.884191e+01 Py = -6.564533e+02 Pz = -3.834036e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 5.122102e+02 Px = 1.194612e+02 Py = -4.980847e+02 Pz = -2.416962e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 2.563147e+03 Px = -1.801128e+03 Py = -1.823639e+03 Pz = -9.220656e-02
[ DEBUG4 "MySiliconTracking_MarlinTrk"] Total 4-momentum of Si tracks : E = 4.050581e+02 Px = -3.805486e+02 Py = -1.387617e+02 Pz = 1.891596e-02
```

# Computing Resource Occupy

- Test with 100 000 muons
  - Close CPU time
  - CEPCSW:
    - Memory leak need to fix



## CPU:

CEPCSW:

User time (s): 706.94

System time (s): 10.38

Marlin:

User time (s): 755.65

System time (s): 4.03

# Future Plans

- Software porting from Marlin/iLCSoft to CEPCSW
  - Algorithms (reconstruction)
  - Geometry management: Gear in Marlin → DD4Hep
- Software improvements
  - Recover the relations between PLCIO data object
- Integration with ACTS, etc.

# Summary

- CEPCSW prototype has been developed using Gaudi, DD4Hep and PLCIO, etc.
- In the prototype,
  - A silicon detector simulation and a Tracking algorithm can be run successfully.
  - Results of CEPCSW and Marlin are intelligible same
  - By implementing data conversion, previously produced MC data can be reused.
- More algorithms are ready to add into the prototype following the given example.
- Future development will be based on Key4HEP collaborating with other future collider experiments

**Thank you !** 25