STCF Parallelized Analysis Framework

Ying Yang, Teng Li, Xingtao Huang

Shandong University

FTCF 2025

2025.11.27

山东大学
SHANDONG UNIVERSITY

Content

01 Motivation

02 Requirement Analysis and Frame Design

03 Examples and Verification

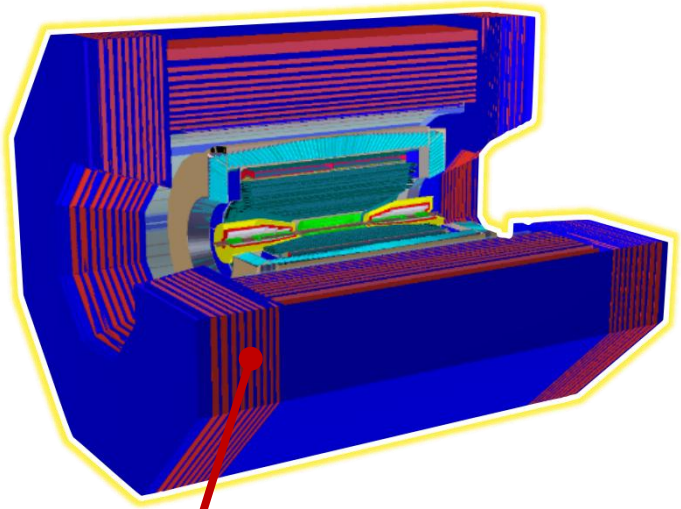
04 Performance

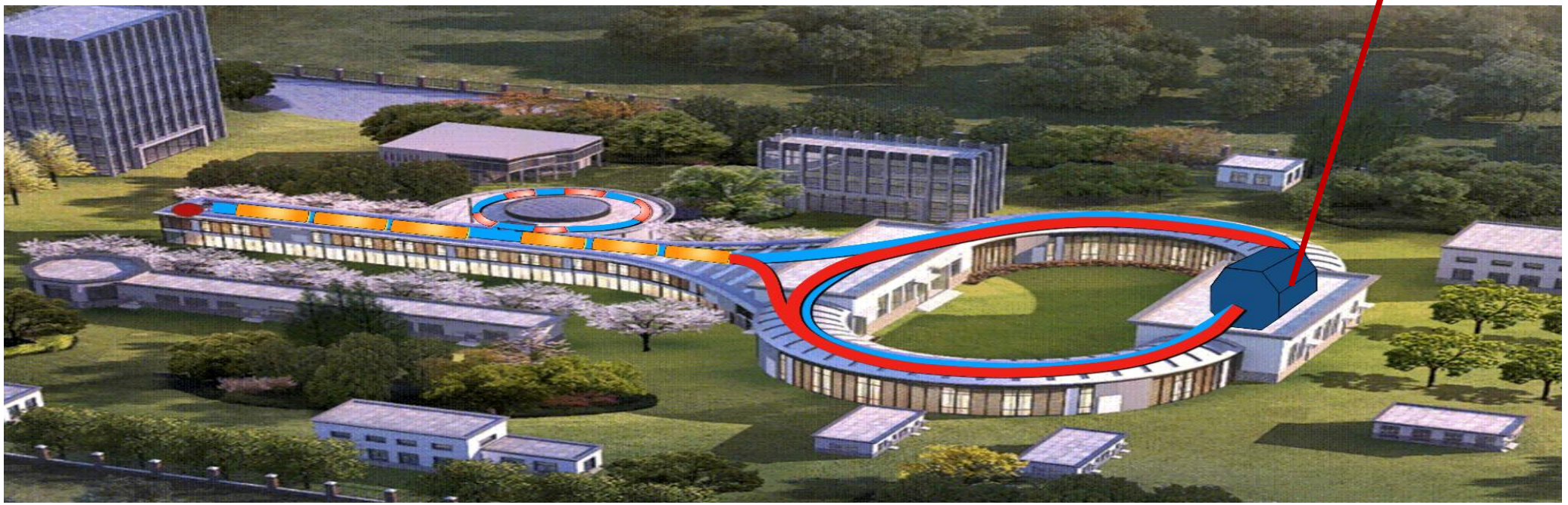
1 Motivation

STCF(Super Tau Charm Facility)

- Wider Energy Range: 2-7 GeV
- Higher luminosity :$\mathbf{0.5}\times \mathbf{10^{35}}\ cm^{-2}\ s^{-1}$

  ~100 higher than BEPCII
- Higher total trigger rate: 400 kHz
- Higher data rate: 30.4 GB/s

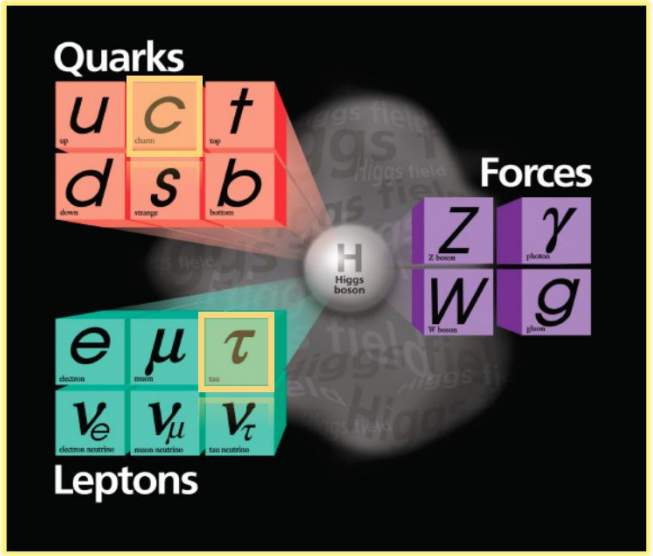




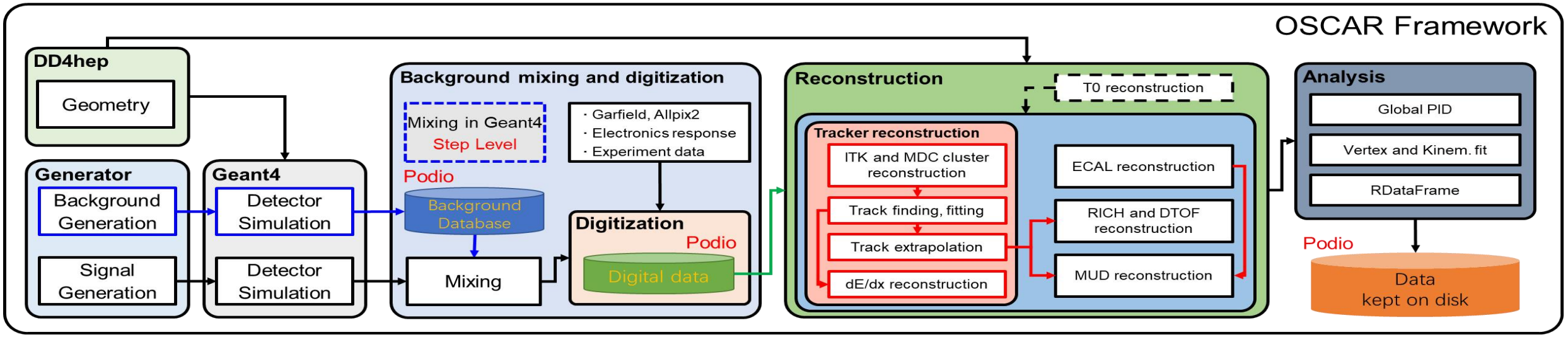
STCF(Super Tau Charm Facility):Physics Goals

Key Physics Goals:

- QCD dynamics and hadron physics
  - Many perturbative QCD properties can be tested
  - Measure strong interactions at the hadron level with great precision, such as time-like nucleon and hyperon form factors
- Electroweak interaction, flavor physics and CP violation
  - The large number of τ pairs produced will provide much more accurate measurements of electroweak couplings and test universality of weak interaction.
- New physics beyond the SM
  - Support searches for τ-lepton LFV (Lepton Flavor Violation)and LNV (Lepton Number Violation)decays with sensitivities of 10^{-8} to 10^{-9}.
  - Serve as a platform to search for proposed new low-mass particles such as dark photons, light scalars and millicharged particles.



OSCAR(Offline Software for Super Tau-Charm Facility)



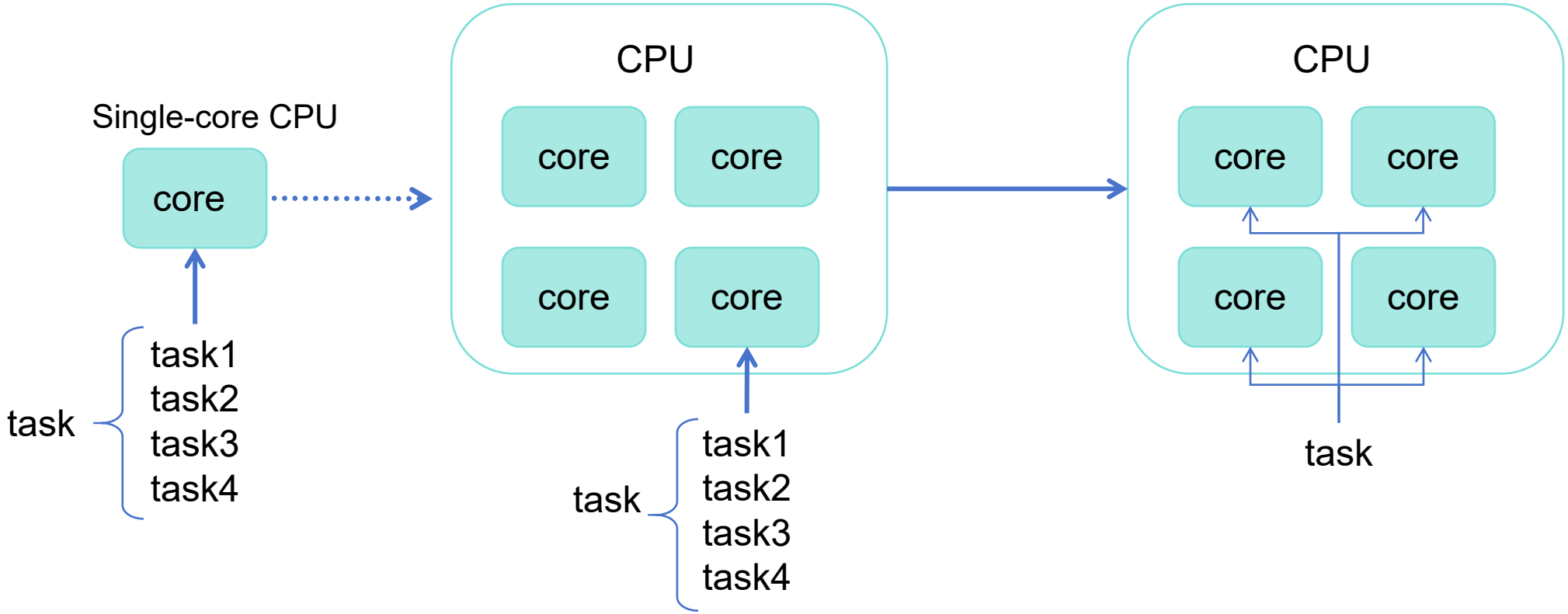
- The whole data processing chain has been built
  - Detector Simulation
  - Background Mixing and digitization
  - Reconstruction
- Physics Analysis as the last but not least step for STCF
  - unprecedented large amount of data to analyze
  - unprecedented precision measurement to achieve

CME (GeV)	Lumi (ab^{-1})	Samples	σ(nb)	No. of Events	Remarks
3.097	1	J/ψ	3400	3.4×10^{12}	
3.670	1	$\tau^+\tau^-$	2.4	2.4×10^9	
3.686	1	$\psi(3686)$	640	6.4×10^{11}	
		$\tau^+\tau^-$	2.5	2.5×10^9	
		$\psi(3686) \to \tau^+\tau^-$		2.0×10^9	
3.770	1	$D^0\bar{D}^0$	3.6	3.6×10^9	
		$D^+\bar{D}^-$	2.8	2.8×10^9	
		$D^0\bar{D}^0$		7.9×10^8	Single tag
		$D^+\bar{D}^-$		5.5×10^8	Single tag
		$\tau^+\tau^-$	2.9	2.9×10^9	
4.009	1	$D^{*0}\bar{D}^0 + c.c$	4.0	1.4×10^9	$CP_{D^0\bar{D}^0} = +$
		$D^{*0}\bar{D}^0 + c.c$	4.0	2.6×10^9	$CP_{D^0\bar{D}^0} = -$
		$D_s^+D_s^-$	0.20	2.0×10^8	
		$\tau^+\tau^-$	3.5	3.5×10^9	

New Opportunity :Evolution of Heterogeneous Computing Architectures

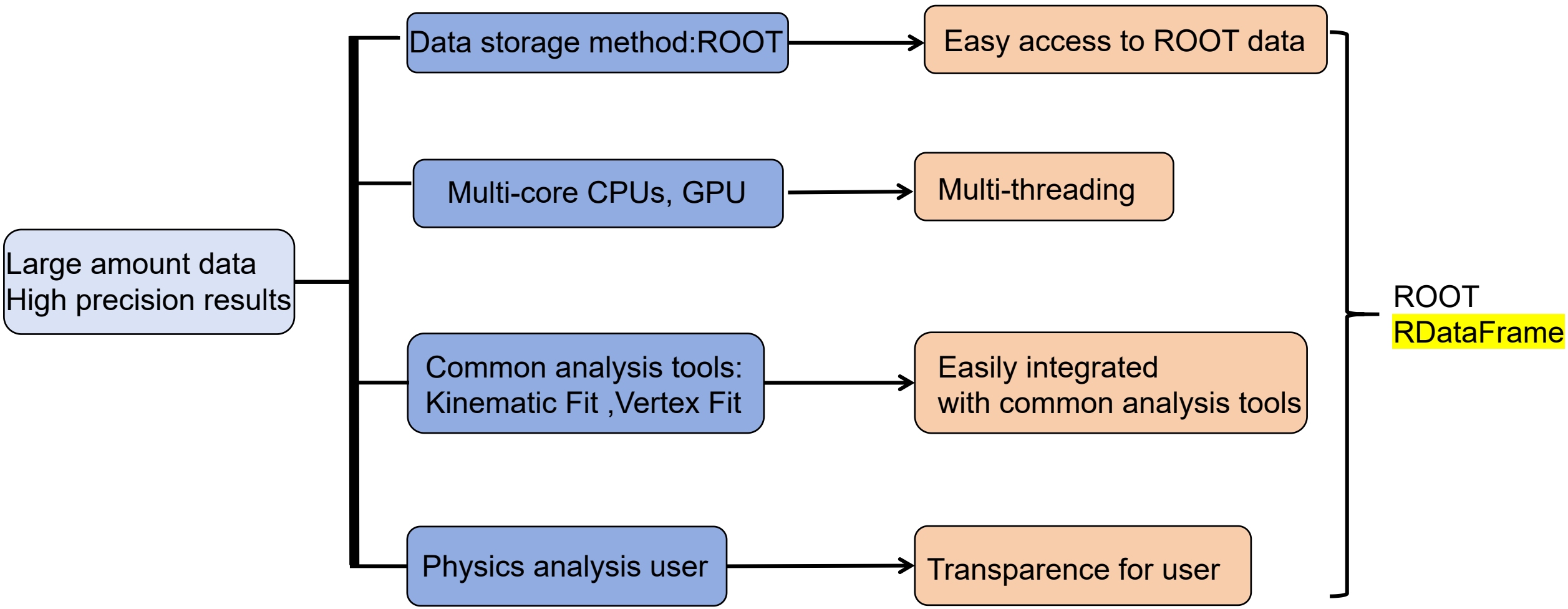
- Computer hardware : single-core CPUs to multi-core CPUs, GPU
- Traditional algorithms :execute one by one ,can not fully utillize computing resource
- Parallel algorithms: execute simultaneously,take advantage of heterogeneous architectures to speed up

➢ **A Parallelized Analysis Framework is essential to efficiently analyze large amount data and achieve high precision results**

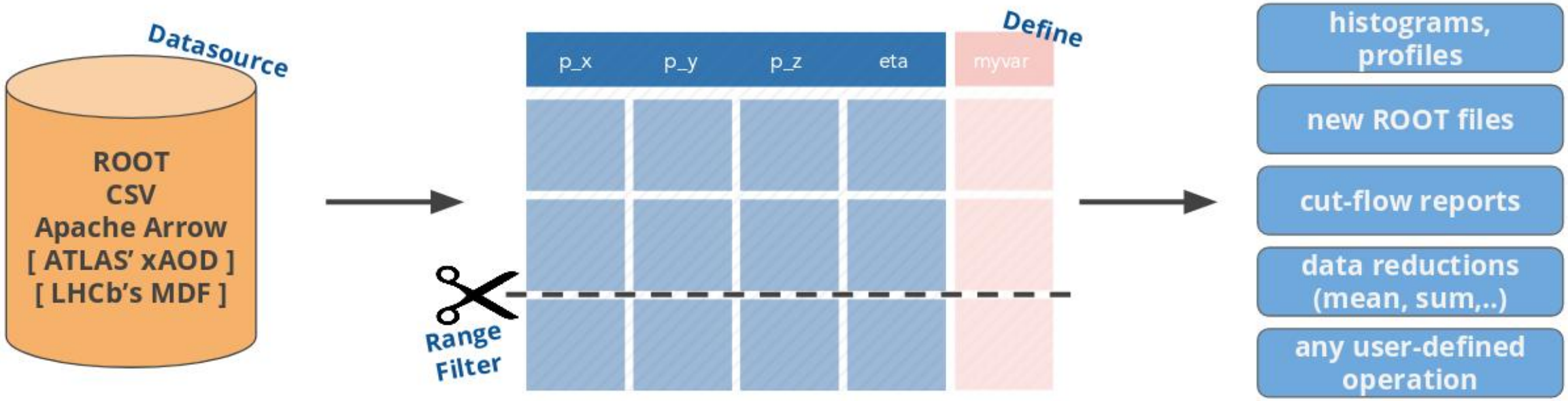


2 Requirement Analysis and Frame Design

Requirement Analysis:What we need for new analysis frame?



ROOT::RDataFrame:Declarative Programming



- RDataFrame supports single TTree and multiple TTrees (i.e., TChain) to read file
- Declarative programming:process data by column
- Traditional function programming

✓ Easy access to ROOT data

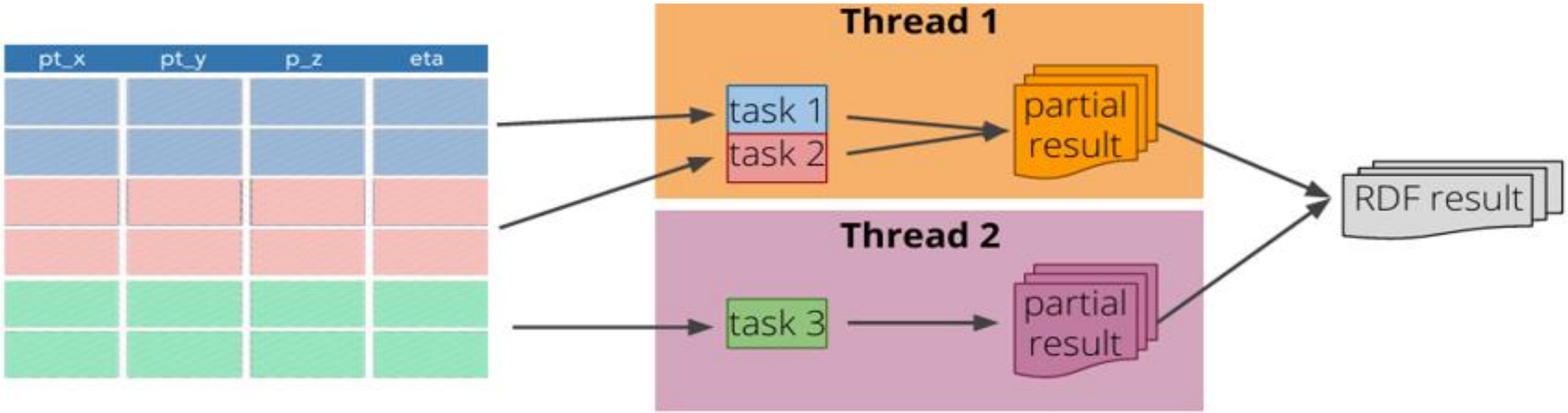
✓ Transparence for user

✓ Easy integration

```
import ROOT
ROOT.gSystem.Load(path/to/myLibrary.so)
ROOT.gInterpreter.Declare('#include "myLibrary.h"')
df=ROOT.RDataFrame("myTree","myroot.root")
newdf=df.Define("p","Complexcalculation(p_x,p_y,p_z)").Filter("p>10").Histo1D("p")
newdf.Draw()
```

```
import ROOT
df=ROOT.RDataFrame("myTree","myroot.root")
newdf=df.Define("p","p_x*p_x+p_y*p_y+p_z*p_z").Filter("p>10").Histo1D("p")
newdf.Draw()
```

ROOT::RDataFrame:Principles of Parallelism



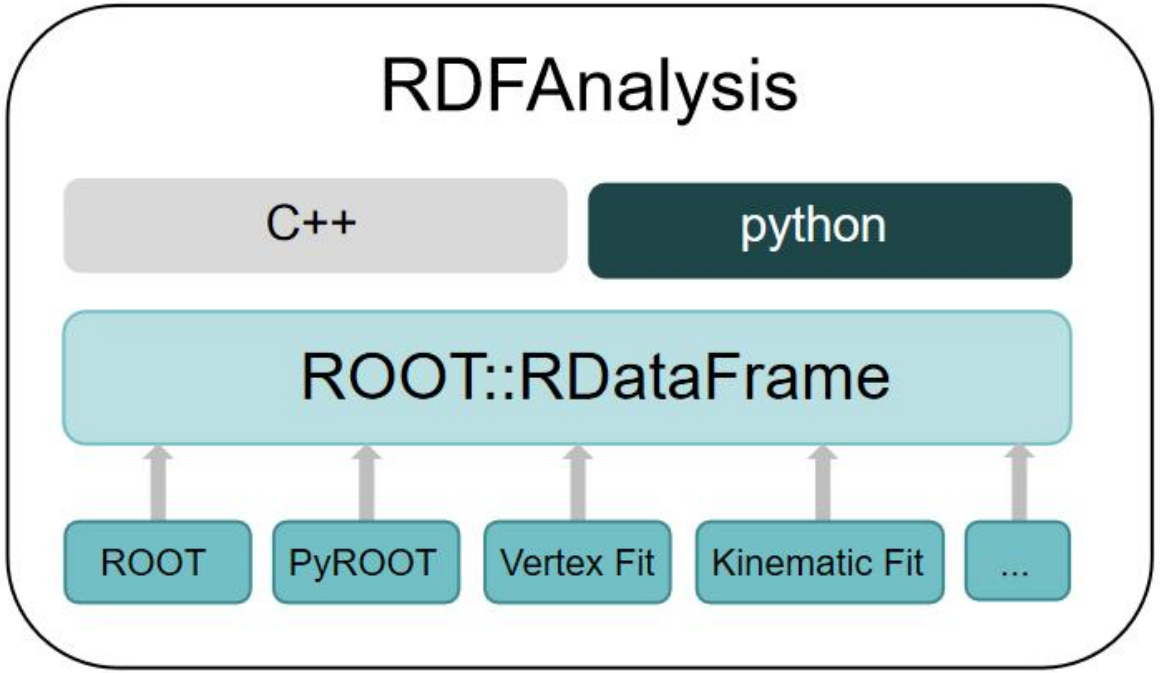
- Divide data into different chunks
- Based on code,generate tasks
- Task scheduling ： Intel Threading Building Blocks (TBB)
- Only a single global switch to turn on parallel computing

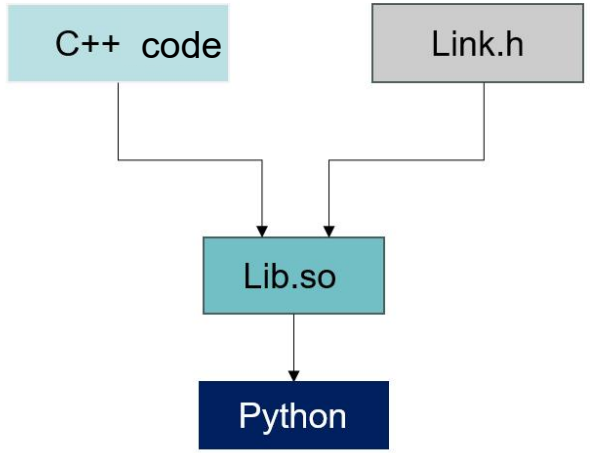
✓Multi-threading

```
ROOT::EnableImplicitMT(); // Enable ROOT's implicit multi-threading
```

Design of RDFAnalysis (RDataFrame analysis framework)

- Programming language: C++ and Python
- Link Physics analysis tools : Vertex Fit ,Kinematic Fit
- Combine declarative programming with traditional function programming





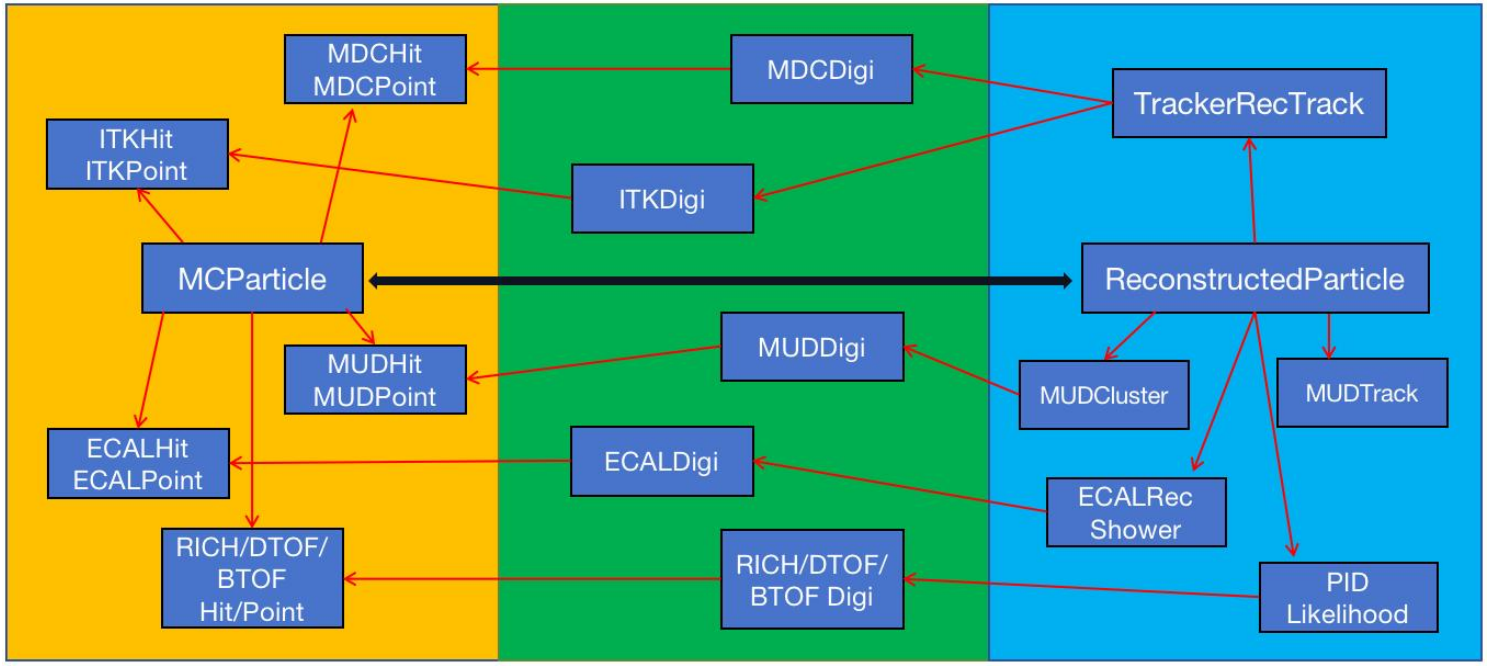
STCF Event Data Model

STCF EDM based on podio, which is also adopted by EDM4hep

- Define event data and relationship between data object in YAML file
- Re-use MCParticle and ReconstructedParticle from EDM4hep as the core index
- Good support for multithreading

Physics Analysis Event Data Model for analysis is building

- Easier access of physics information



YAML

```
MCParticle:
  Description : "Data class for storing Monte Carlo particles"
  Author : "SDU"
  Members :
    - int trackID                // track ID
    - int PDG                    //PDG code of the particle
    - int generatorStatus        //status of the particle as defined by the generator
    - int simulatorStatus        //status of the particle from the simulation program - use BIT constants
    - int type                   //particle type. only generatorStatus== 1 or 2 has the type
    - float charge               //particle charge
    - float time                 //creation time of the particle in [ns] wrt. the event, e.g. for preassig
    - double mass                //mass of the particle in [MeV]
    - Vector3d vertex            //production vertex of the particle in [mm].
    - Vector3d endpoint          //endpoint of the particle in [mm]
    - Vector3f momentum          //particle 3-momentum at the production vertex in [GeV]
    - Vector3f momentumAtEndpoint //particle 3-momentum at the endpoint in [GeV]
    - Vector3f spin              //spin (helicity) vector of the particle.
```

code generator

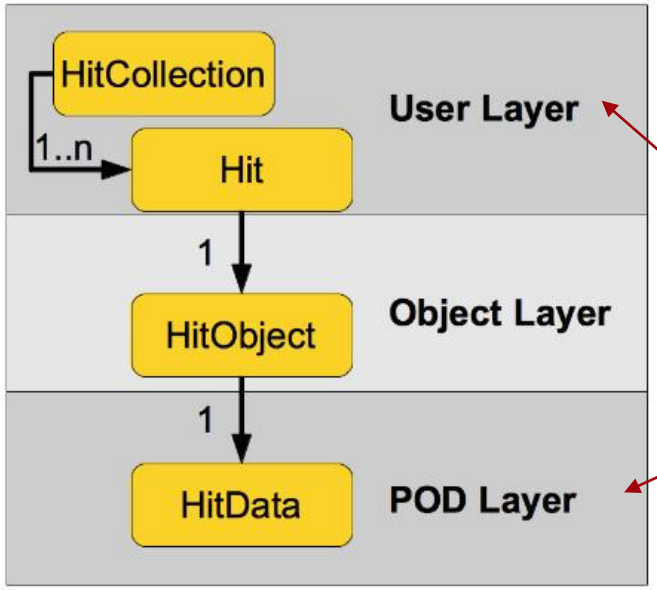
C++

MCParticle.h	MCParticle.cpp
MutableMCParticle.h	MutableMCParticle.cpp
MCParticleObj.h	MCParticleObj.cpp
MCParticleData.h	MCParticleData.cpp
MCParticleCollection.h	MCParticleCollection.cpp
MCParticleCollectionData.h	MCParticleCollectionData.cpp

Based on YAML definition, generate EDM C++ code accordingly

RDFAnalysis Access to Data Model

- User Layer:lightweight handles to objects and collections
- Object Layer:transient objects (HitObject), which handle relations between objects of EDM and also manage POD objects
- POD Layer:actual data structures



- OSCAR(podio)：User layer
- RDFAnalysis:POD layer

OSCAR(podio)	RDFAnalysis
TrackerRecTrack atrack; TrackerHypoTrack ahypotrack=atrack.getTrackHypos(n);	TrackerRecTrackData atrack; ROOT::VecOps::Rvec<TrackerHypoTrackData> TrackerHyposCol; TrackerHypoTrackData ahypotrack=TrackerHyposCol[atrack.trackHypos_begin+n];

Interfaces Code in OSCAR

Vertex Fit

```
01  HepPoint3D vx(0., 0., 0.);
02  HepSymMatrix  Evx(3, 0);
03  double bx = 1E+6;
04  double by = 1E+6;
05  double bz = 1E+6;
06  Evx[0][0] = bx * bx;
07  Evx[1][1] = by * by;
08  Evx[2][2] = bz * bz;
09
10  //Create initial vertex information
11  VertexParameter vxpar;
12  vxpar.setVx(vx);
13  vxpar.setEvx(Evx);
14
15  VertexFit *vtxfit00 = VertexFit::instance();
16  vtxfit00->init();
17  //Add tracks
18  vtxfit00->AddTrack(0,mproton,PpHypoTrk,PpTrk);
19  vtxfit00->AddTrack(1,mpion,PimHypoTrk,PimTrk);
20  //Add vertex
21  vtxfit00->AddVertex(0,vxpar,0,1);
22  if(!(vtxfit00->Fit(0)))  return  true;
```

Convert the reconstructed data model to WTrackParameter

- use AddTrack to get Vertex Fit and KalmanKinematicFit

Kinematic Fit

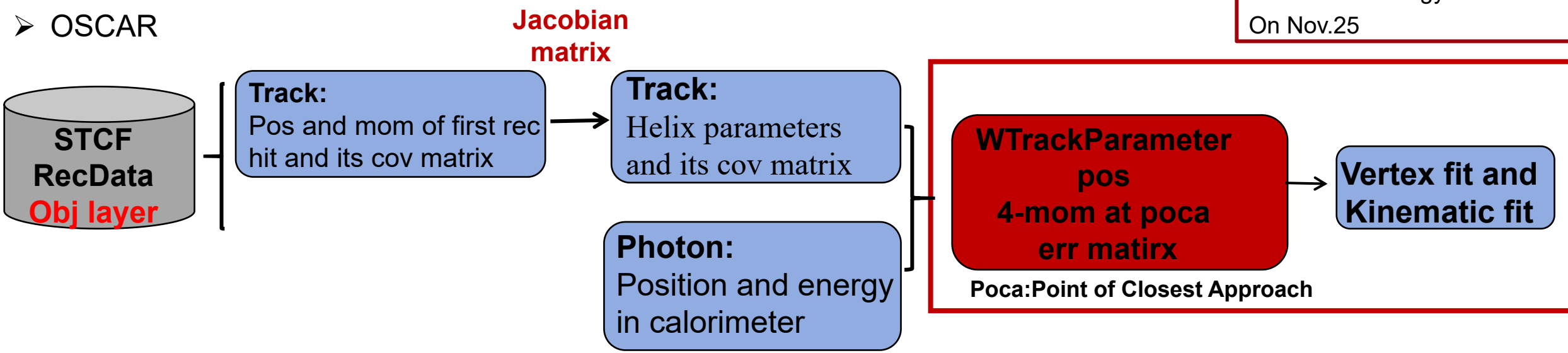
```
01  KalmanKinematicFit * kmfit = KalmanKinematicFit::instance();
02  kmfit->init();
03  //Add terminal tracks
04  kmfit->AddTrack(0,wpip);
05  kmfit->AddTrack(1,wpim);
06  kmfit->AddTrack(2,wp);
07  kmfit->AddTrack(3,wpm);
08  //Add constraint
09  kmfit->AddFourMomentum(0, 3.097);
10  //Fit
11  bool okfit = kmfit->Fit();
12  if(!okfit) return true;
```

More details in Mingyu's talk On Nov.25

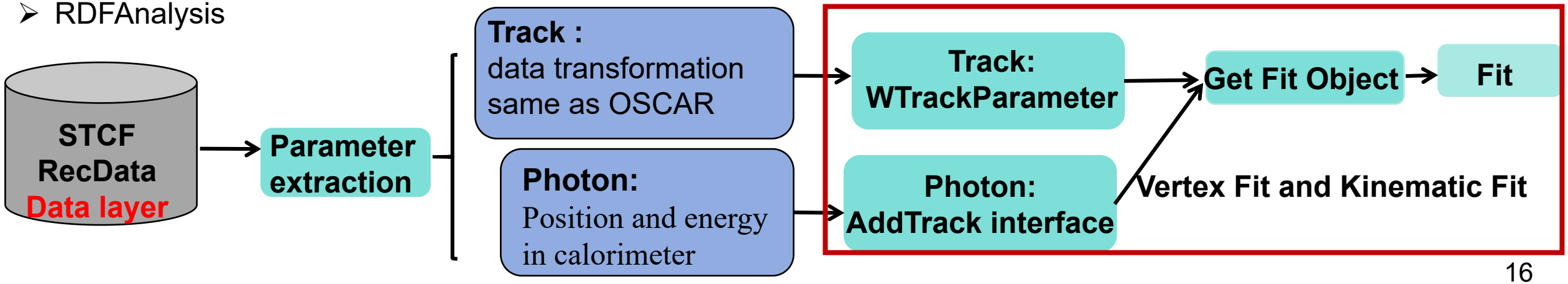
Interfaces Design in OSCAR and RDFAnalysis

More details about OSCAR interfaces in Mingyu's talk On Nov.25

- OSCAR



- RDFAnalysis



Interfaces in RDFAanalysis

- Track selection interface

```
select_chargeTrack(
        const ROOT::VecOps::RVec<TrackerRecTrackData> &track, //all track
        const ROOT::VecOps::RVec<Vector3d> &momtrack, //all track momentum
        const ROOT::VecOps::RVec<int> &iGoodtrk,          // good track  index
        const int n,                                      // particle id   (0-4 : e、μ、π、K、p )
        const float min1p,                                // maximum momentum
        const float max1p,                                // minimum momentum
        const int charge )                                // particle charge
```

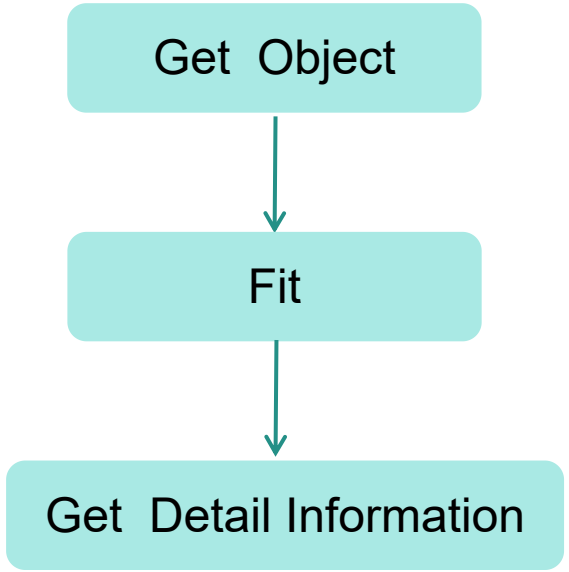
Vertex Fit and Kinematic Fit Interface

- Vertex Fit interface

```
get_vfitObject(const WTrackParameter &wtrk1,
               const WTrackParameter &wtrk2)// two tracks
getvfit_Massafter(const VertexFit&  vtxfit)//from VertexFit Object get mass
```

- Kinematic Fit interface

```
KalmanKinematicFitObject(const float &Ecms,//energy of CMS
const ROOT::VecOps::RVec<WTrackParameter>&iwtrks, // charge tracks
const ROOT::VecOps::RVec<RecECALShowerData> &igtrks //neutral tracks
 )(int nctrk,int nntrack) // number of charge track ,number of neutral particles
// get KalmanKinematicFit Object
```



3 Examples and Verification

Analysis Codes and Python scripts for $J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$

- select tracks

```python
import ROOT
ROOT.gSystem.Load("libRDFAnalysis.so")
#Dynamic Link Library

load=ROOT.f()
#Load function and class
ROOT.EnableImplicitMT(nthread)
#Turn on multithreading

df=ROOT.RDataFrame("events","Rec_rhopi.root")
#Read data

#select charge tracks
df1=df.Define("wtrk",f"VertexFitUtil::SelectUtil::get_2wtrk_byWLpid(GlobalWLPidCol,TrackerRecTrackCol,TrackerHypoTrackCol,{PDGID},{particle_ID},{cutxy},{cutz})")\
      .Filter(f"wtrk[0].size()=={nct/2}&&wtrk[1].size()=={nct/2}")\
      .Define("pipwtrk","wtrk[0][0]")\
      .Define("pimwtrk","wtrk[1][0]")

#select neutral tracks
df2=df1.Define("nwtrk",f"VertexFitUtil::SelectUtil::select_Neutral_Tracks(RecECALShowerCol,RecExtTrackCol_4,RecExtTrackCol,{particle_nfID},{cut_dang})")\
       .Filter(f"nwtrk.size()>={nnt}")

#vertex fit for charge tracks
#match charge tracks and neutral tracks
df3=df2.Define("vertexfit","VertexFitUtil::get_vfitObject(pipwtrk,pimwtrk)")\
       .Filter("vertexfit.Fit(0)")\
       .Redefine("vertexfit","VertexFitUtil::update_vfitData(vertexfit)")\
       .Define("kwtrk","VertexFitUtil::get_forkfit(vertexfit)")\
       .Define("goodnwtrk","VertexFitUtil::SelectUtil::select_Neuwtrk_forkfit(nwtrk,kwtrk[0],kwtrk[1])")\
       .Filter(f"goodnwtrk.size()=={nnt}")
#kinematic fit
#get data
df4=df3.Define("kfit",f"KalmanKinematicFitObject(3.097,kwtrk,goodnwtrk)({nct},{nnt})")\
       .Filter("kfit->Fit()")\
       .Define("m_mrho_before","VertexFitUtil::get_massbeforekfit(kfit,0,1)")\
       .Define("m_mrho_after","VertexFitUtil::get_massafterkfit(kfit,0,1)")\
       .Define("m_mpi0_before","VertexFitUtil::get_massbeforekfit(kfit,2,3)")\
       .Define("m_mpi0_after","VertexFitUtil::get_massafterkfit(kfit,2,3)")
#store file

df4.Snapshot("events","rhopi0.root",["m_mrho_before","m_mrho_after","m_mpi0_before","m_mpi0_after"])

```

$J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$

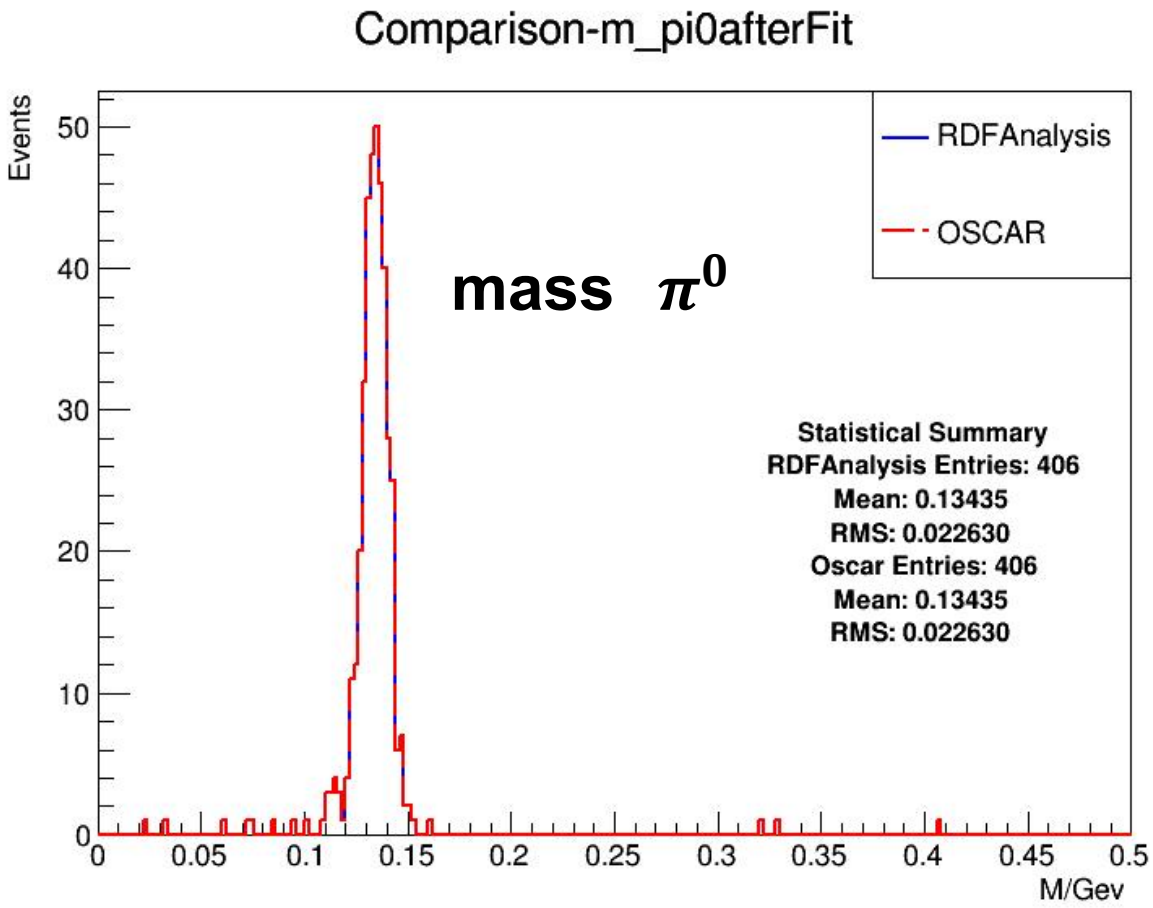
- User Page

```
# 1. set input or output
input_file = "Rec_rhopi.root"
output_file = "analysis_results_rhopi.root"
nthread=8
 # 2. set
 particle_config = {
      'cutxy':10,              #cut xy
      'cutz':20,               #cut z
      'cut_dang':25,           #cut min angle
      'particle_ID': 2,        #select  particle ID
      'particle_nfID':2,       #particle ID of neutracl track vertex
      'PDGID':211,             #select PDGID
      'nnt':2,                 #number of charge tracks
      'nct':2                  #number of neutral track


 }

# output
output_vars = ["m_mrho_before","m_mrho_after","m_mpi0_before","m_mpi0_after"]
```

set

- ➢ cut condition
- ➢ number of thread
- ➢ particle information

- Result



Analysis Codes and Python scripts for $J/\psi \to \Lambda\bar{\Lambda} \to p\pi^-\bar{p}\pi^+$

```
#select wtrk
#by cut condition
#momentum range of particles
#get good track index

df1=df.Define("Goodtrk",
           f"VertexFitUtil::SelectUtil::select_iGoodtrk(TrackerR
           ecTrackCol,TrackerHypoTrackCol,TrackerRecTrackCol_1,
           {particle1_ID},{cutxy},{cutz},{cutcostheta})")\
      .Filter(f"Goodtrk.size() >= {min_trk}")\
      .Define("ncharge_track","VertexFitUtil::SelectUtil::select
           _nctrack(TrackerRecTrackCol,Goodtrk)")\
     .Filter(f"ncharge_track[0]>={min_pos}&&ncharge_track[1]>={m
           in_neg}")\
      .Define("pimtrk",f"VertexFitUtil::SelectUtil::select_charg
           eTrack(TrackerRecTrackCol,TrackerRecTrackCol_1,Goodtr
           k,{particle1_ID},{P1min},{P1max},-1)")\
      .Define("piptrk",f"VertexFitUtil::SelectUtil::select_charg
           eTrack(TrackerRecTrackCol,TrackerRecTrackCol_1,Goodtr
           k, {particle1_ID},{P1min},{P1max},1)")\
#match good tracks and get WTrackParameter
dfwtrk=df1.Define("pp_pimwtrk",f"VertexFitUtil::SelectUtil::sele
                  ct_2WTP(TrackerRecTrackCol,TrackerHypoTrackCol,
                  pptrk,pimtrk,{particle1_ID}, {particle2_ID})")\
         .Define("pm_pipwtrk",f"VertexFitUtil::SelectUtil::selec
                           t_2WTP(TrackerRecTrackCol,TrackerHypo
                           TrackCol, pmtrk,piptrk,{particle1_ID},
                           {particle2_ID})")\
         .Filter("pm_pipwtrk.size()==2&&pp_pimwtrk.size()==2")
```

```
#Vertex Fit
#Seconde Vertex Fit
#get data
df2=dfwtrk.Define("Vertexfit_p_pi","VertexFitUtil::get_vfitObject(
                                    pp_pimwtrk[0],pp_pimwtrk[1])
                                    ")\
          #get Vertexfit Object
         .Filter("Vertexfit_p_pi.Fit(0)","lamdvfit")\#fit
         .Redefine("Vertexfit_p_pi","VertexFitUtil::update_vfitDa
                                    ta(Vertexfit_p_pi)")\
        #update Vertexfit Object
         .Define("wtrkppi","VertexFitUtil::get_forkfit(Vertexfit_
                                    p_pi)")\
          #get tracks for Kinematic Fit
         .Define("RDFAm_lambeforefit","VertexFitUtil::getvfit_Mas
                                    sbefore(Vertexfit_p_pi)")\
         .Define("RDFAm_lamaftervfit","VertexFitUtil::getvfit_Mas
                                    safter(Vertexfit_p_pi)")\
         .Define("secondvfit","VertexFitUtil::get_svfitObject(Ver
                                texfit_p_pi)")\
         .Filter("secondvfit.Fit()","lamdsvfit")\
         .Define("RDFAlam_decaylength","VertexFitUtil::getsvfit_D
                                    ecaylength(secondvfit)")
#Kinematic Fit
df4=df3.Define(f"kfit","KalmanKinematicFitObject(3.097,wtrkppi,Verte
                                xfit_pbar_pip)({nct},{nnt})")\
       .Filter("kfit->Fit()","kfit")\
       .Define("RDFAm_lamafterkfit","VertexFitUtil::get_massafterkfi
                                    t(kfit,0,1)")\
```

User Page $J/\psi \to \Lambda\bar{\Lambda} \to p\pi^-\bar{p}\pi^+$

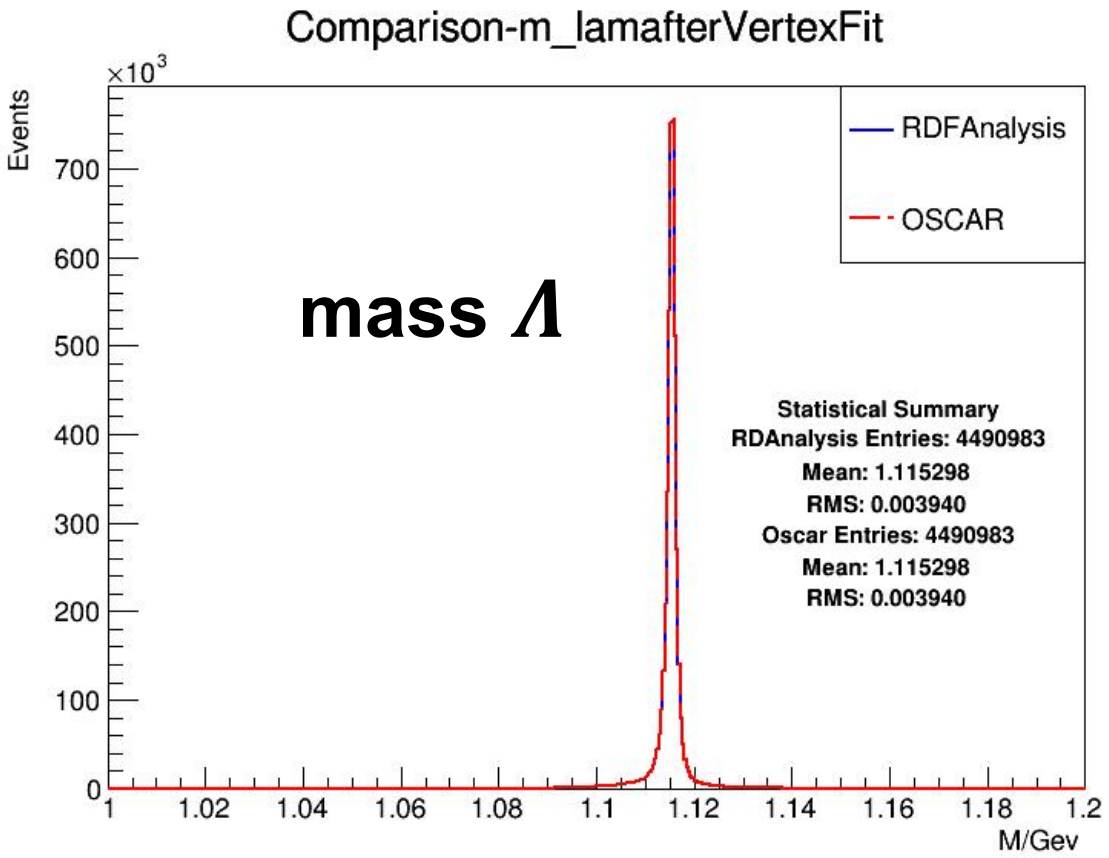
```
# set input or output
input_file = "reclam20w_mgp15.root"
output_file = "analysis_results.root"
nthread=8
# set
particle_config = {
    'cutxy':10.0
    'cutz':30.0
    'cutcostheta':0.93    # cut
    'particle1_ID': 2,    # select  particle1 ID
    'particle2_ID': 4,      # select  particle2 ID
    'min_positive_charge': 2,    # number of positively charged tracks
    'min_negative_charge': 2,     # number of negatively charged tracks
    'P1min':0,
    'P1max':0.5,
    'P2min':0.5,
    'P2max':10000

}

# output information
output_vars = ["RDFAm_lamaftervfit","RDFAm_lamafterkfit","RDFAm_lambeforefit","chi2","RDFAantilam_decaylength","RDFAlam_decaylength","RDFAm_antilambeforefit","RDFAm_antilamafte
rvfit","RDFAm_antilamafterkfit"]
```

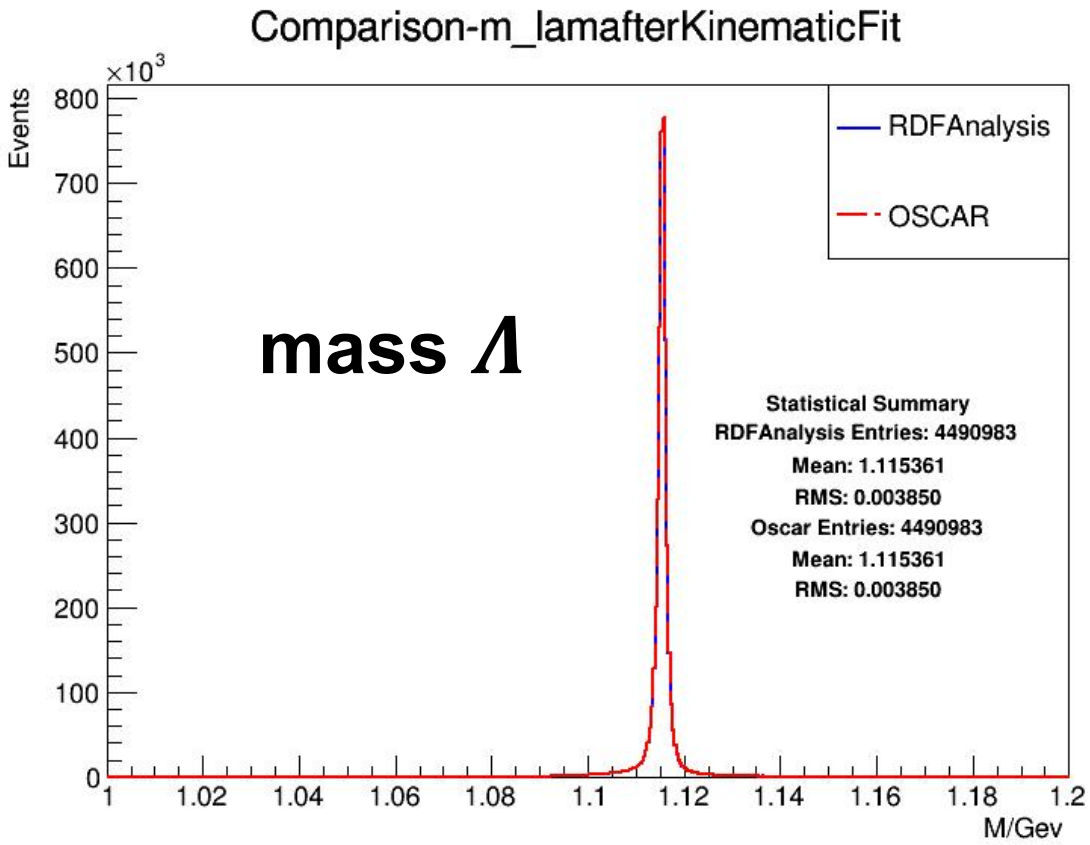
user set：

- input file
- output file
- number of thread
- information of particle
- cut condition
- output information

$J/\psi \rightarrow \Lambda\bar{\Lambda} \rightarrow p\pi^{-}\bar{p}\pi^{+}$

10 million events





4 Performance

Performance Analysis

- $J/\psi \rightarrow \Lambda\overline{\Lambda} \rightarrow p\pi^{-}\bar{p}\pi^{+}$:
  200 thousand events

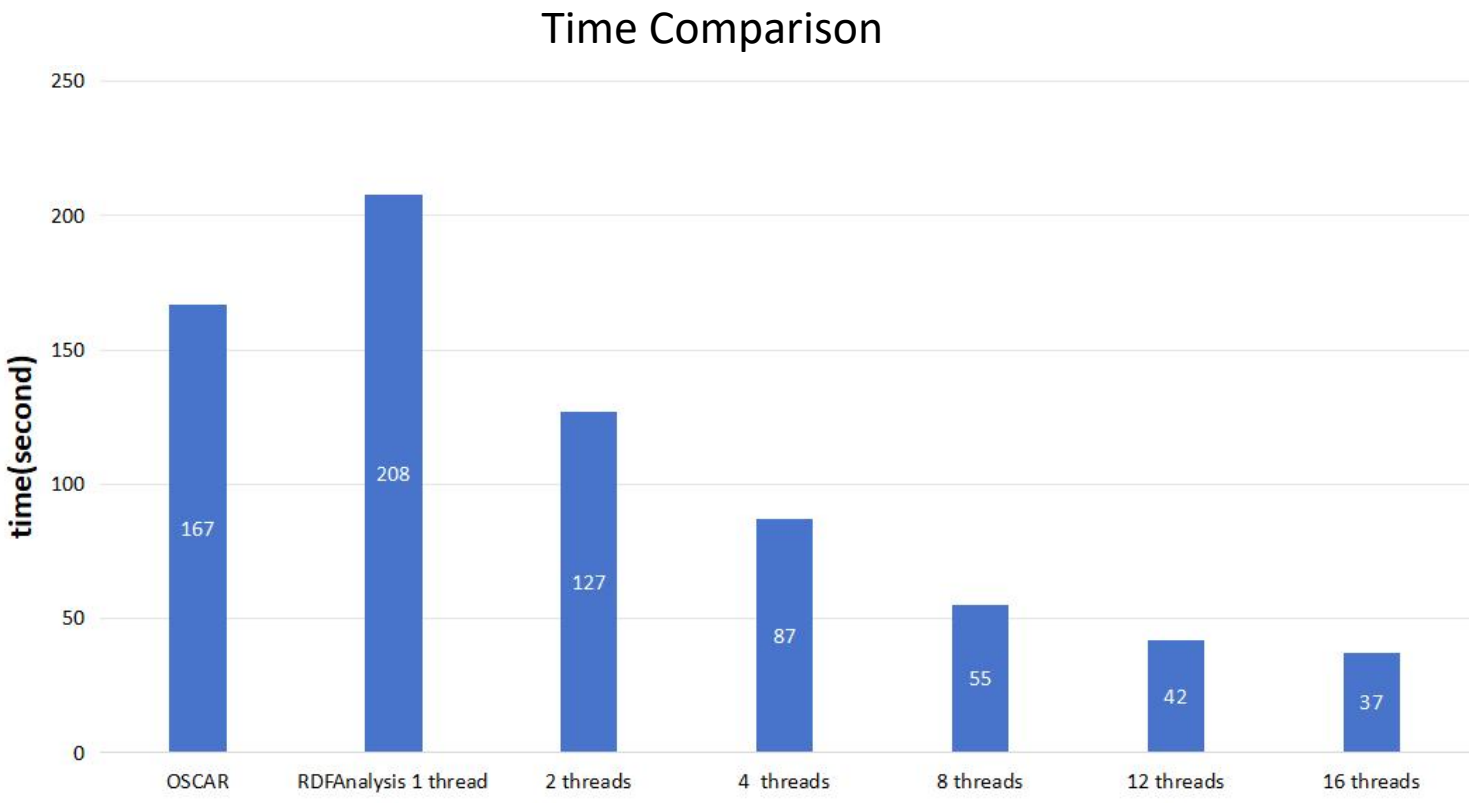
- Reasons for not meeting expectations:

➢ on 8 threads , one third of time is used in malloc call

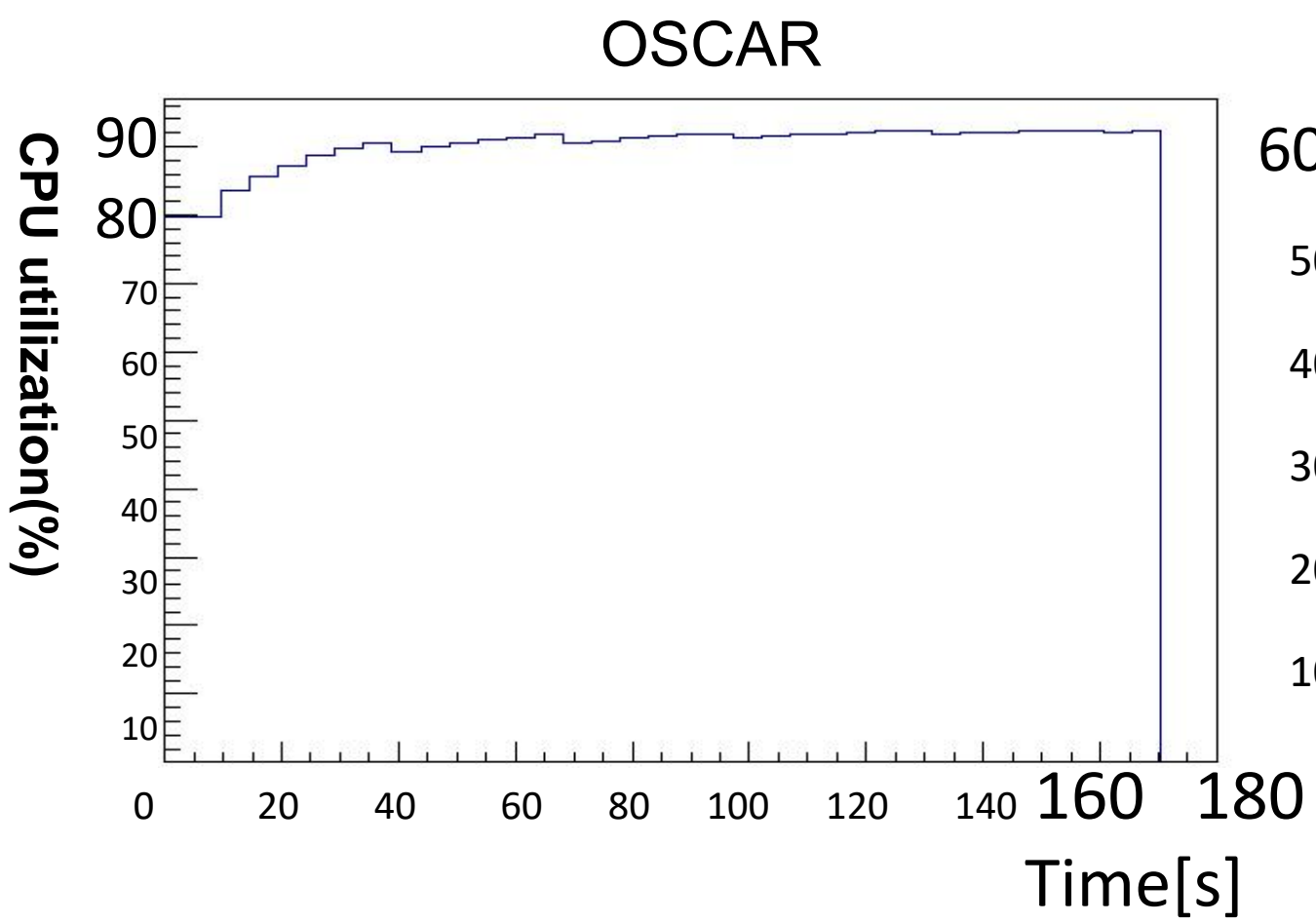
➢ too much intermediate variable

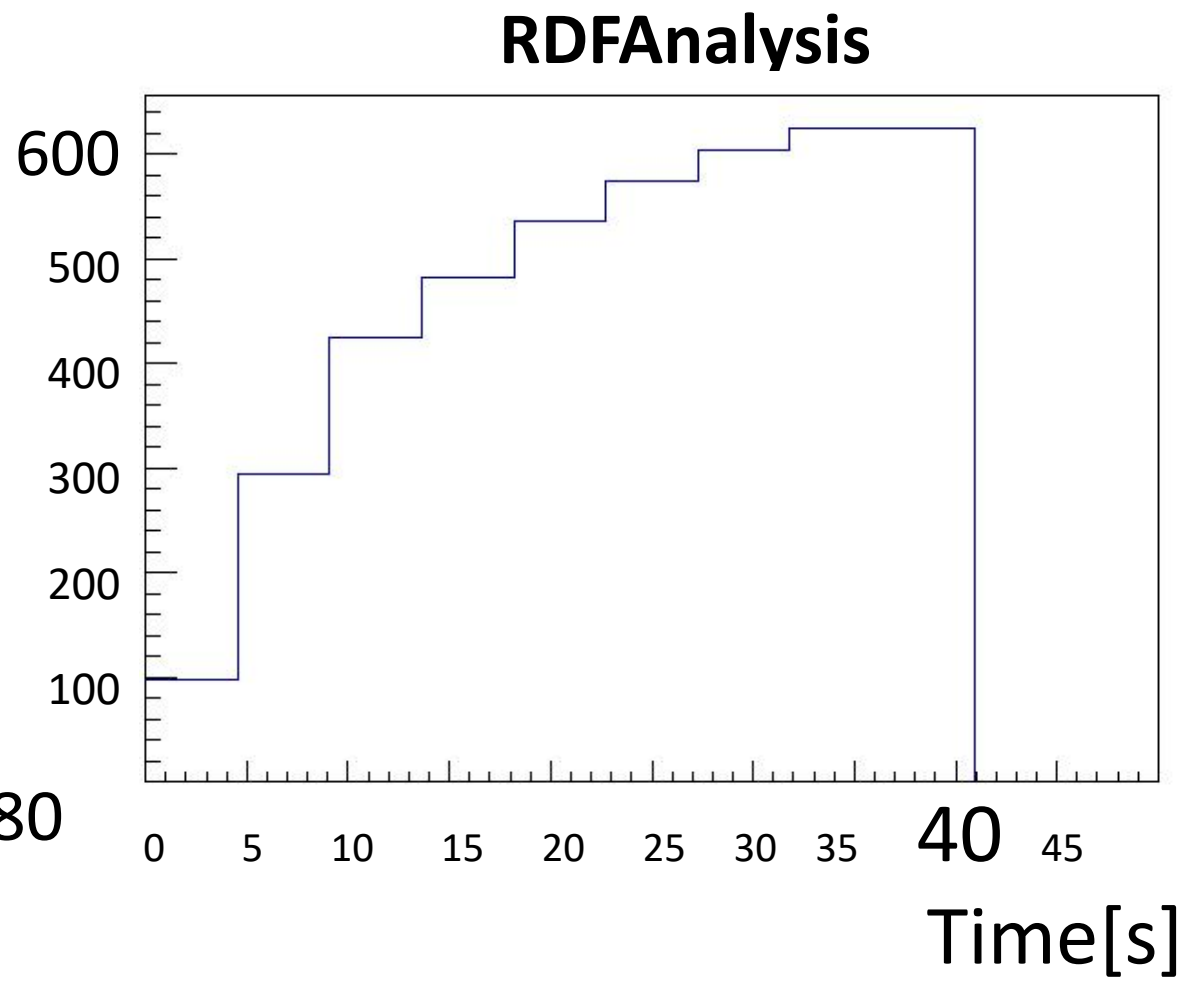
➢ Amdahl's Law

$$t = t_{serial} + \frac{t_{parallel}}{n}$$



Comparison of **CPU utilization** under OSCAR and RDFAnalysis





Summary

- A Parallelized Analysis Framework is designed and implemented for STCF
  - Use parallel multithreading :speed up physics analysis ;full use of computing resources
  - Combine declarative programming with traditional function programming to improve code readability
  - Vertex Fit and Kinematic Fit tools have been added
  - Correctness check: $J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$

$$J/\psi \to \Lambda\overline{\Lambda} \to p\pi^-\bar{p}\pi^+$$

What to do next

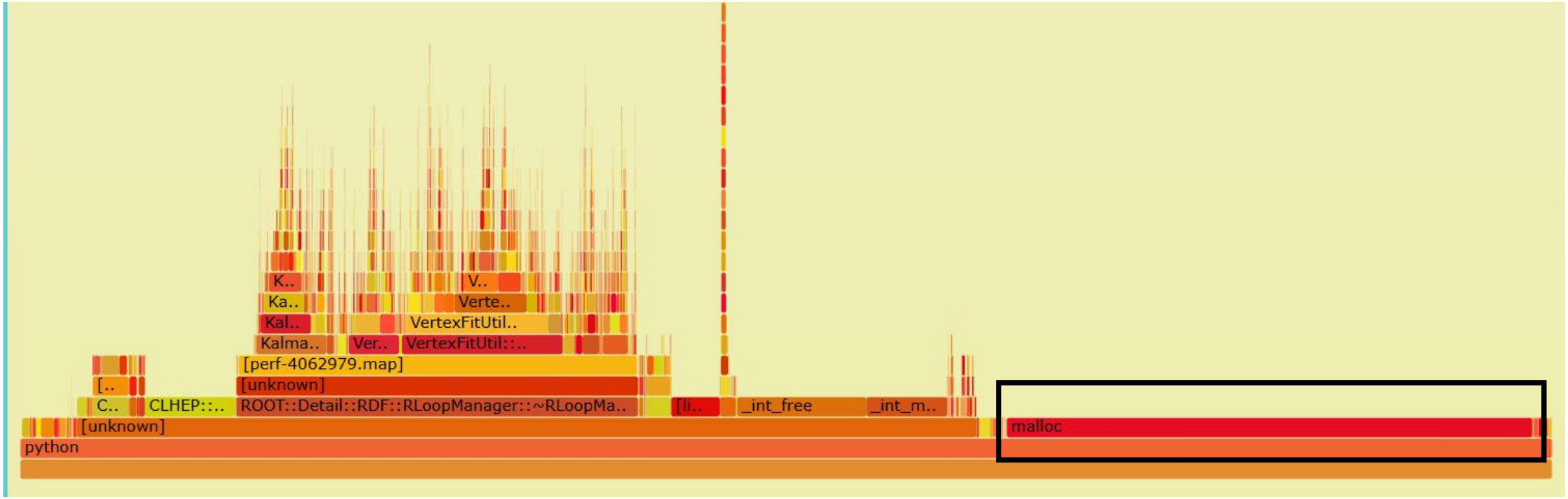
- Interface optimization
- Adopted with Physics Analysis EDM
- Add more analysis tools : Global Vertex Fit tool, D-tagging tool, ...

THANKS!

Reporter:Ying Yang

Date： 2025.11.27

Pref CPU Flame Graph (8 threads)



proportion:34.3% 323e9