STCF基于RDataFrame并行物理分析框架

杨莹　李腾　黄性涛

山东大学

STCF 2026

2026.7.2

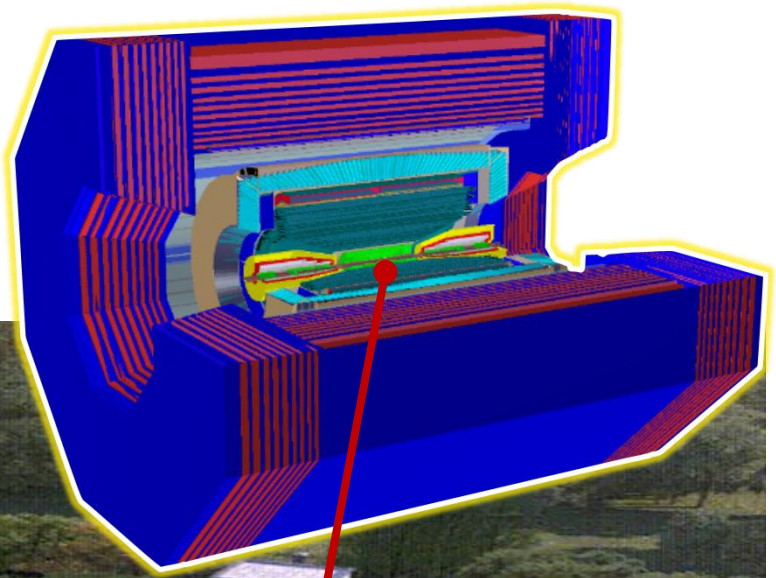
山东大学 SHANDONG UNIVERSITY

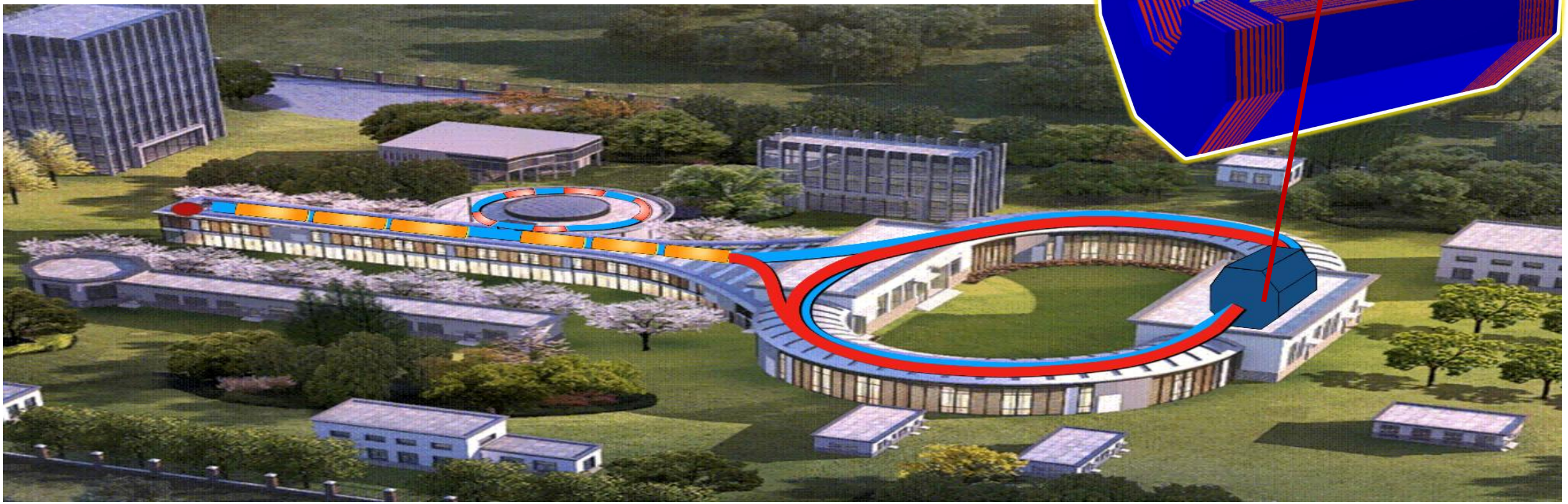
目录

1 研究背景

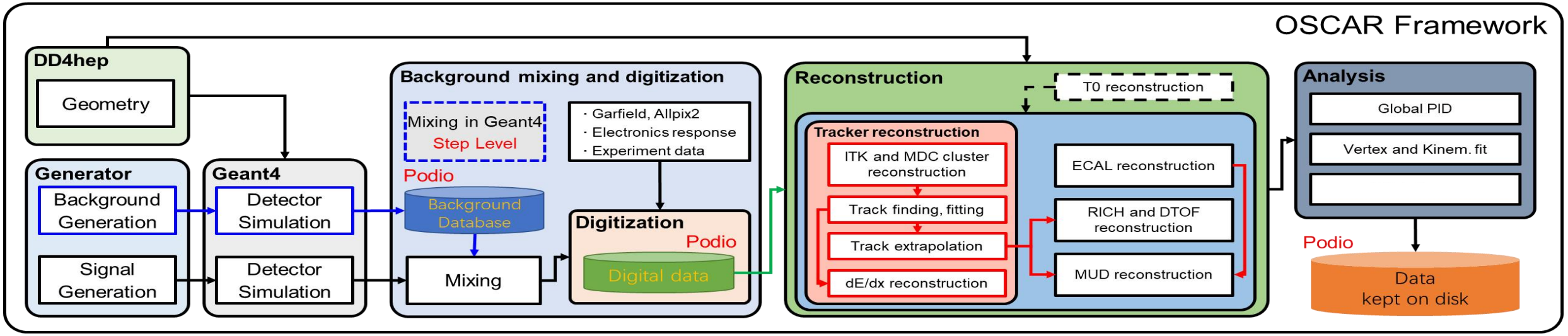
STCF(Super Tau Charm Facility)

- **物理目标：**精确测量τ轻子、粲夸克相关物理过程。
- **核心参数：**能量范围覆盖 2 - 7 GeV，峰值亮度高达**0.5**× $\mathbf{10^{35}}$ $cm^{-2}s^{-1}$。
- **数据挑战：**海量数据实时产生，每年原始数据量达**数百PB**级。





OSCAR(Offline Software for Super Tau-Charm Facility)



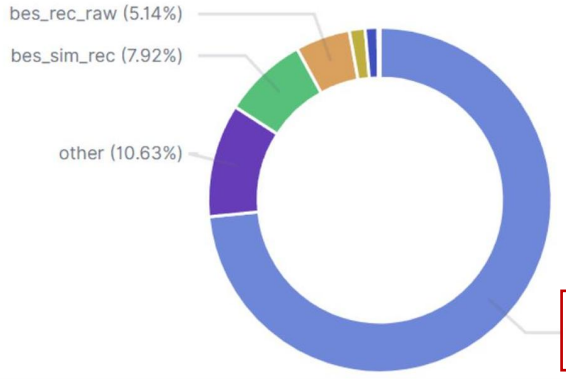
OSCAR：

- 覆盖从探测器模拟、本底混合数字化到数据重建的完整离线处理链路。

- **核心挑战一：物理分析中海量数据的高效处理**
  - 传统串行模式失效，需构建高并发并行分析框架，突破I/O瓶颈与算力调度限制，保障物理分析数据处理时效性。

- **核心挑战二：高精度物理结果**
  - 顶点拟合等算法计算密集，需优化底层实现与资源分配，在有限算力下确保结果的精度与可靠性。

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	



2024年BESIII计算资源使用情况

BES 图片来源：
https://indico.ihep.ac.cn/event/22334/contributions/157038/attachments/78289/97385/BESIII_Offline_Software_Chengdu_202405.pdf

2 物理需求和框架设计

需求分析

挑战与目标

挑战：海量数据

面对PB级实验数据，处理效率亟待提升

目标：高精度结果

确保物理分析的准确性与可重复性

存储兼容

无缝处理ROOT数据文件

硬件适配

多核CPU与GPU：充分利用并行算力

软件集成

分析工具复用：灵活接口

用户易用

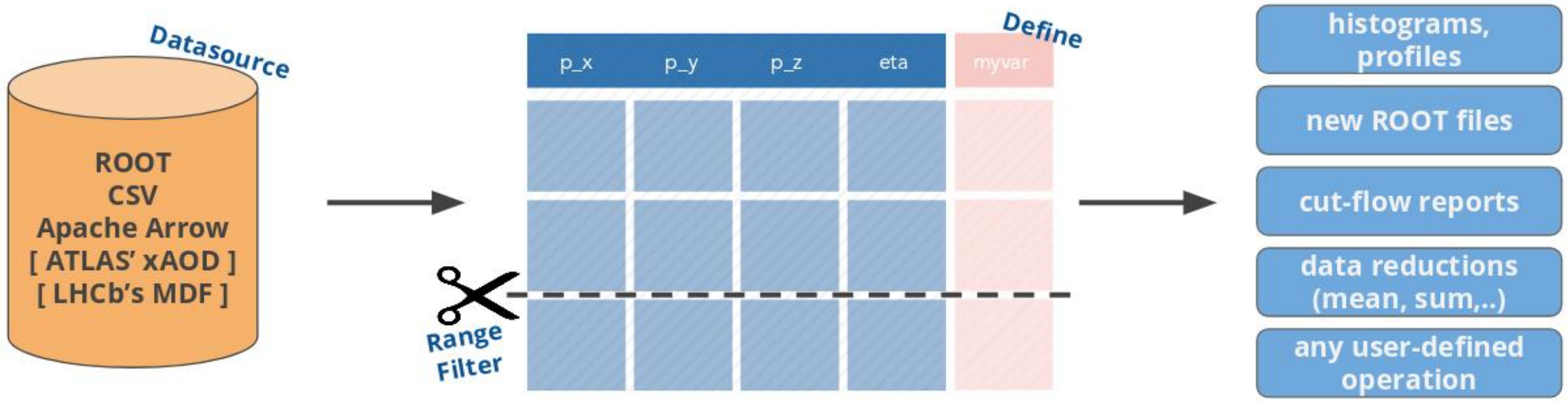
极简交互：透明化处理

技术选型

ROOT::RDataFrame

CERN ROOT库中的模块

ROOT::RDataFrame:声明式编程



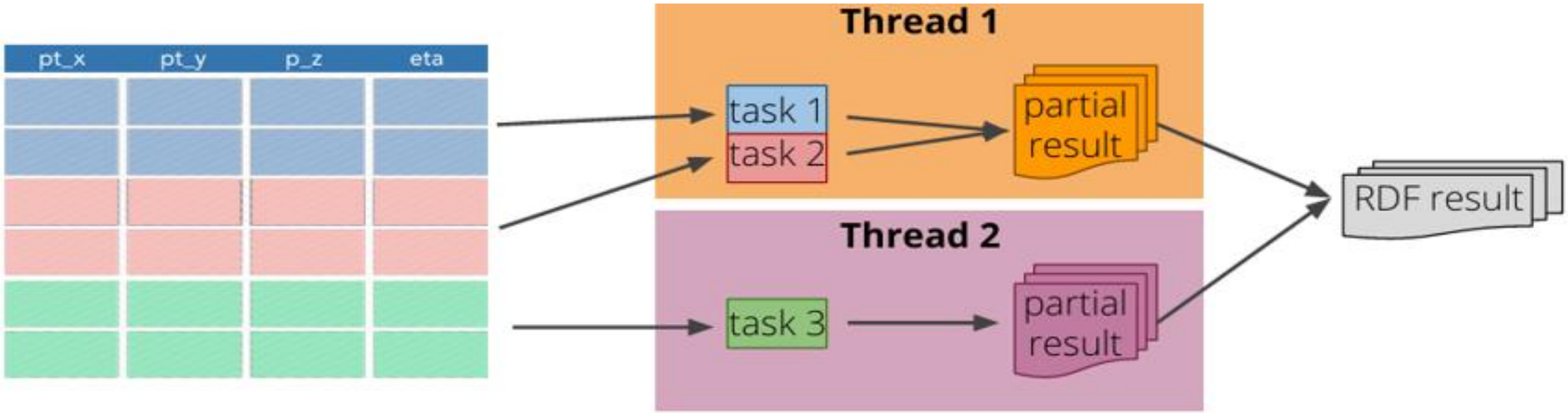
- **声明式编程模型：**用户只需描述数据分析的目标（“做什么”），无需关心底层数据遍历、循环控制等具体实现（“怎么做”）。
- 传统函数编程

```
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()
```

- ✓ 便捷访问 ROOT 数据
- ✓ 用户透明
- ✓ 便于集成

ROOT::RDataFrame:并行实现



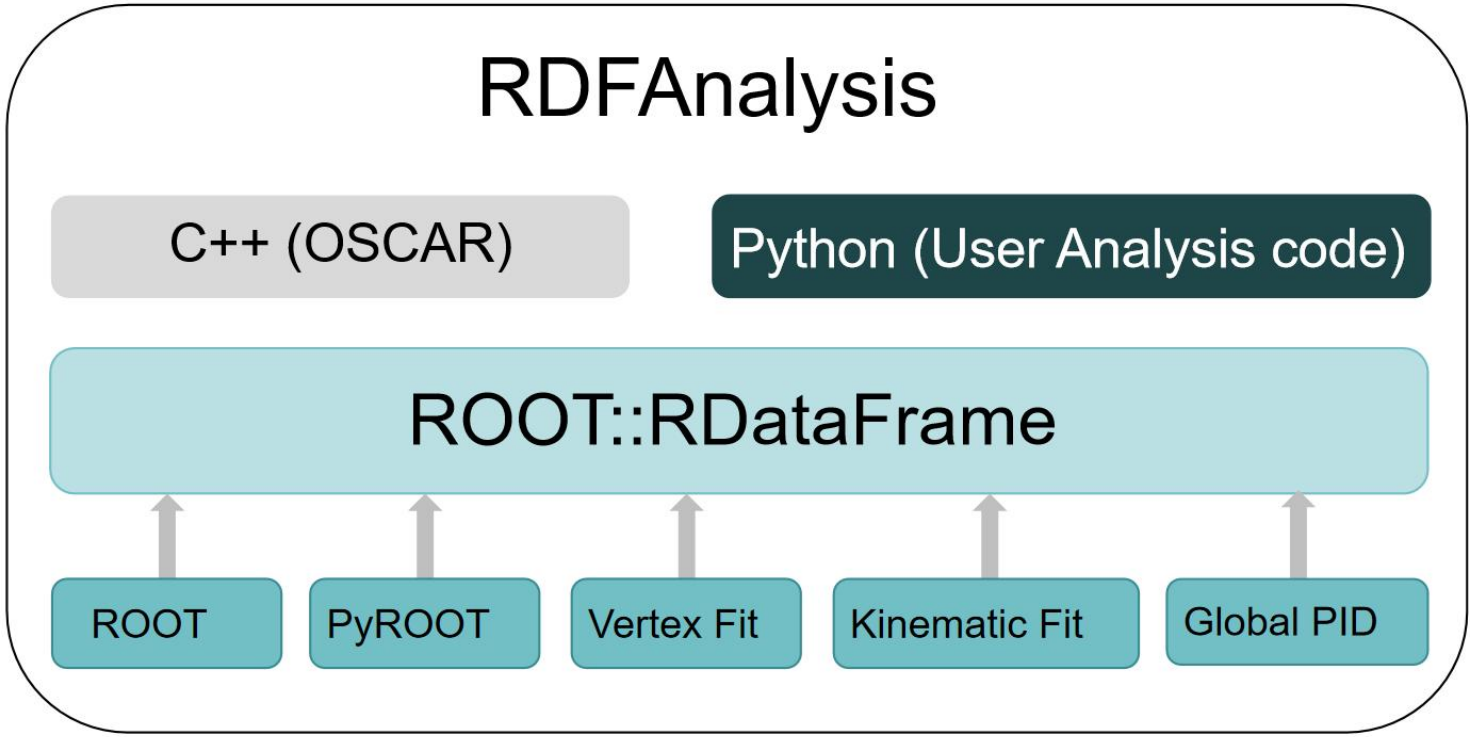
- **Intel TBB 内核调度：**内部集成Intel TBB并行库，实现任务的动态拆分与智能调度。
- **多核自动并行化：**任务自动分发至CPU多核心执行，充分利用硬件资源提升计算效率。
- **零并行代码成本：**用户无需掌握并行编程，现有代码自动获得并行加速能力。

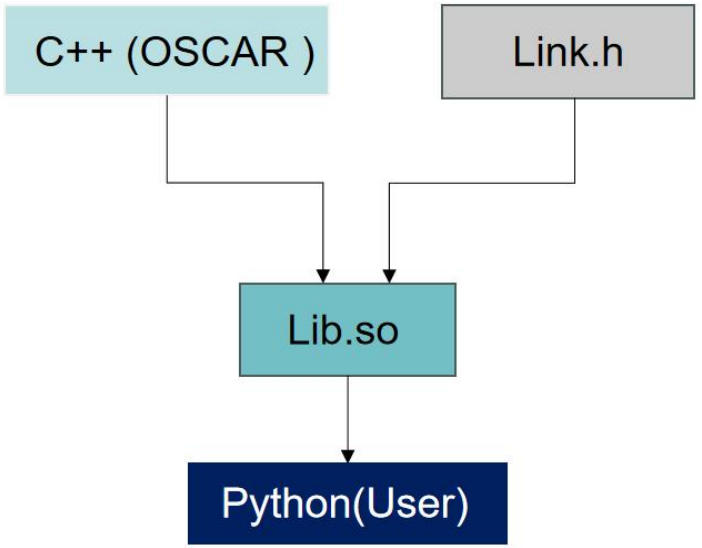
✓多线程并行

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

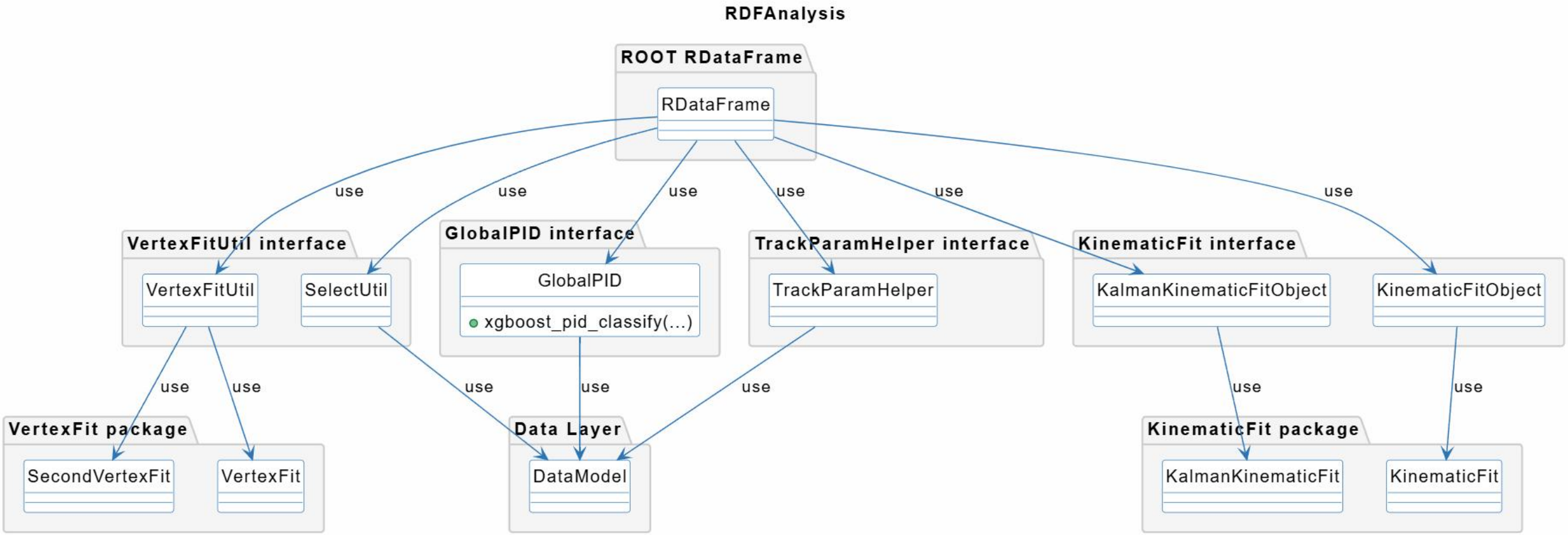
RDFAnalysis (RDataFrame analysis framework)

- **双语言支持：** 同时提供高性能 C++ 底层接口与简洁易用的 Python 上层接口，兼顾计算效率与开发灵活性。
- **核心物理工具集成：** 封装 Vertex Fit、Kinematic Fit、GlobalPID 核心高能物理分析工具。
- **混合编程范式兼容：** 完美融合传统函数式编程与现代声明式编程风格，适配不同开发者场景需求。





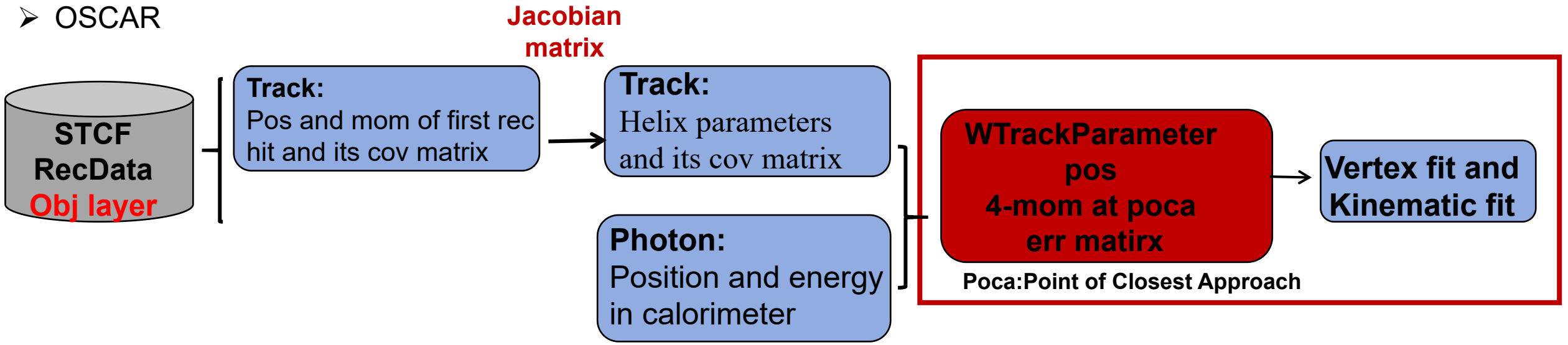
RDFAnalysis 接口设计



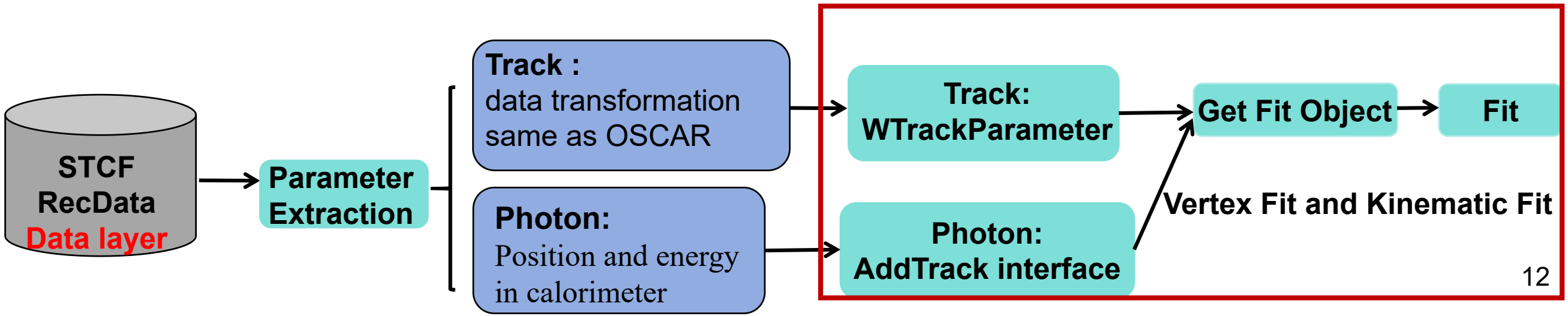
- **TrackParamHelper**：封装径迹数据结构，提供便捷的径迹参数访问，简化物理量调用流程。
- **VertexFitUtil**：集成顶点拟合核心算法，提供用户接口。
- **KinematicFit**：实现运动学约束拟合功能，提供用户接口。
- **GlobalPID**：特征值提取，融合多种探测器信息，获得五粒子鉴别概率。

RDFAnalysis 接口设计: KinematicFit VertexFit

- OSCAR

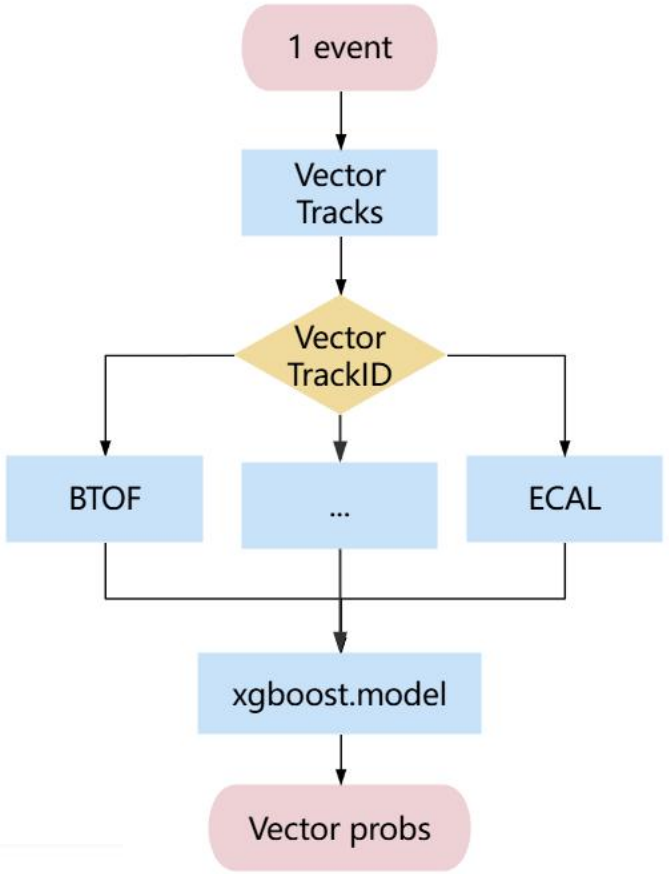


- RDFAnalysis



RDFAnalysis 接口设计：GlobalPID

- 在探测器数据中，每条径迹都有一个唯一的 `TrackID`
- GlobalPID 通过 `TrackID` 将不同探测器的数据关联起来
- 获取每条Track对应的dEdX, BTOF，DTOF，MUD，ECAL，Track信息
- 提取特征值传入模型
- 得到PID鉴别结果：5粒子概率



用户使用示例

- 传入需要做PID鉴别的径迹
- 定义好径迹条件
- 应用条件到径迹中

```
df2= build_pid_chain("Tracks",df1)

GooTrk=df2.Define("GoodTrackCondition","pi_prob > k_prob && pi_prob > mu_prob")\
          .Define("GoodTrack","Tracks[GoodTrackCondition]")
```

3 应用举例与验证

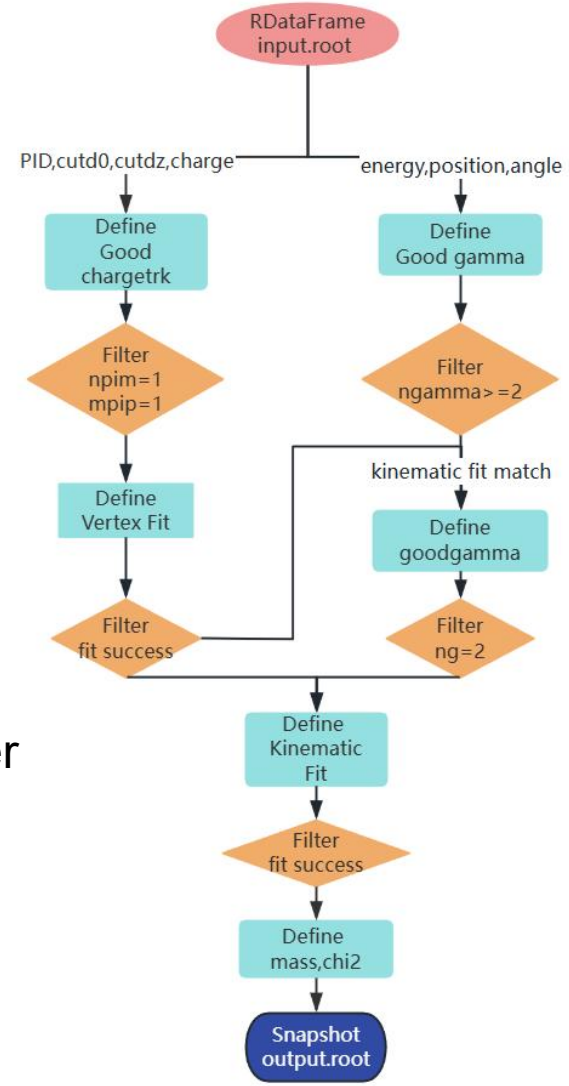
分析算法示例：$J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$

核心代码片段

```
# 1. 环境配置与初始化
config = {"input_file": "...", "model_path": "...", "thread":4}
set_model_path(config["model_path"])
ROOT.EnableImplicitMT(config["thread"])
# 2. 数据读取 (RDataFrame)
df = ROOT.RDataFrame("events", config["input_file"])
# 3. 径迹选择 & PID分析
df1 = build_track_chain(df, 2)
df2 = build_pid_chain("TrackerRecTrackCol", df1)
# 4. 好径迹筛选 (π⁺π⁻)
df3 = df2.Define("GoodPimtrk", "...pi_prob > k_prob...")\
         .Define("GoodPiptrk", "...pi_prob > k_prob...")\
         .Define("Pip", "TrackerRecTrackCol[GoodPiptrk]")\
         .Define("Pim", "TrackerRecTrackCol[GoodPimtrk]")\
         .Filter("Pim.size() == 1 && Pip.size() == 1")
# 5. 顶点与运动学拟合
df4 = df3.Define("pimwtrk", get_wtrack("Pim[0]", 2))\
         .Define("pipwtrk", get_wtrack("Pip[0]", 2))

df6 = df4.Define("vertexfit", get_vfit("pipwtrk", "pimwtrk"))\
         .Filter("vertexfit.Fit(0)")
df7 = df6.Define("kfit", "KinematicFitObject(3.097, ...)")\
         .Filter("kfit->Fit()")
# 6. 结果输出
df7.Snapshot("events", "output.root", ["m_mrho_afterfit", ...])
```

- Global PID
- 设置筛选径迹条件
- 应用筛选径迹条件
- 数据类型转换 WTrackParamter

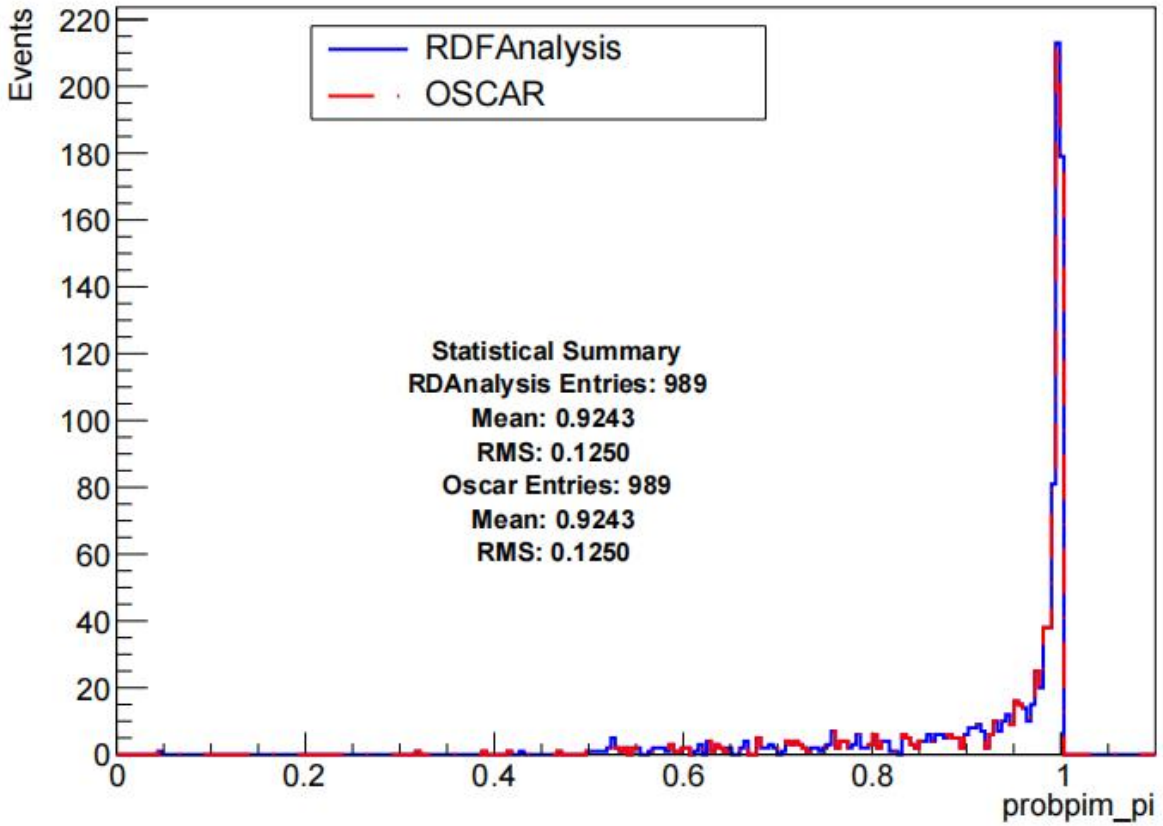
物理分析流程图



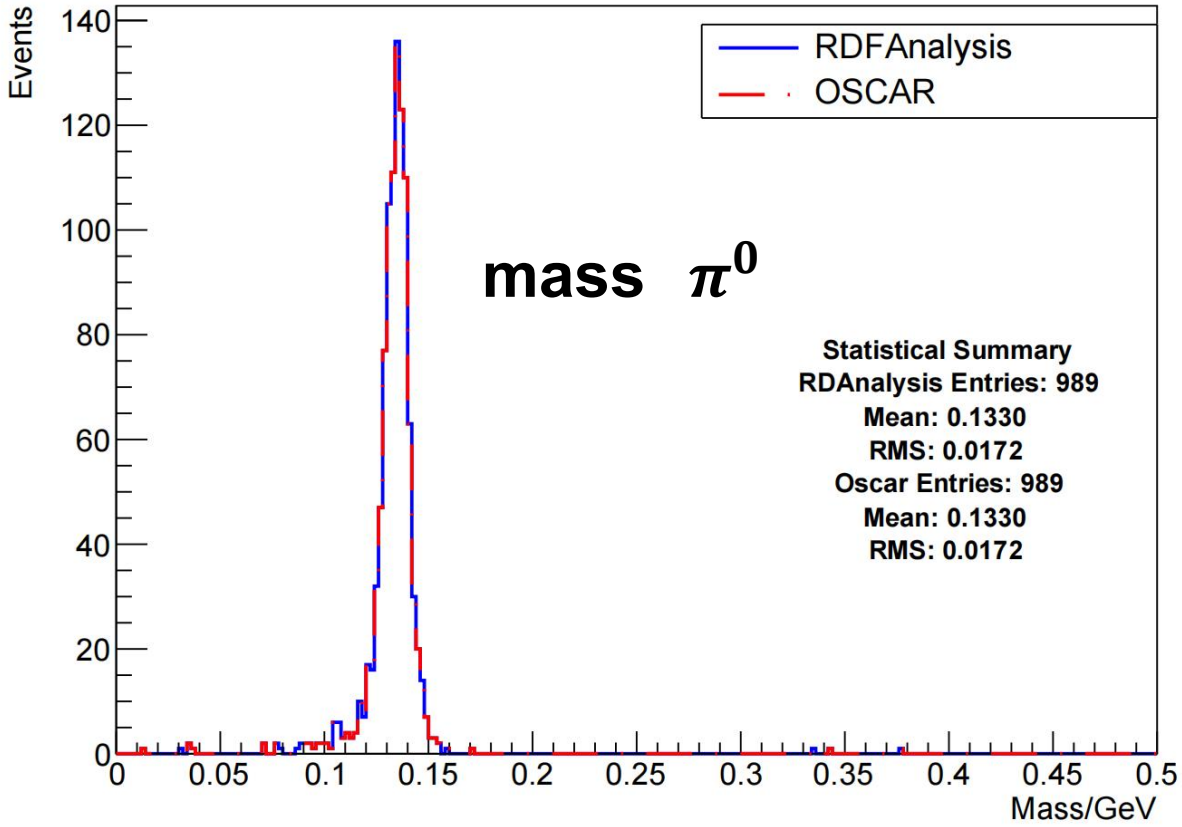
https://git.ustc.edu.cn/oscar1/rdfanalysis

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

- 挑选的π^-粒子为π的概率值



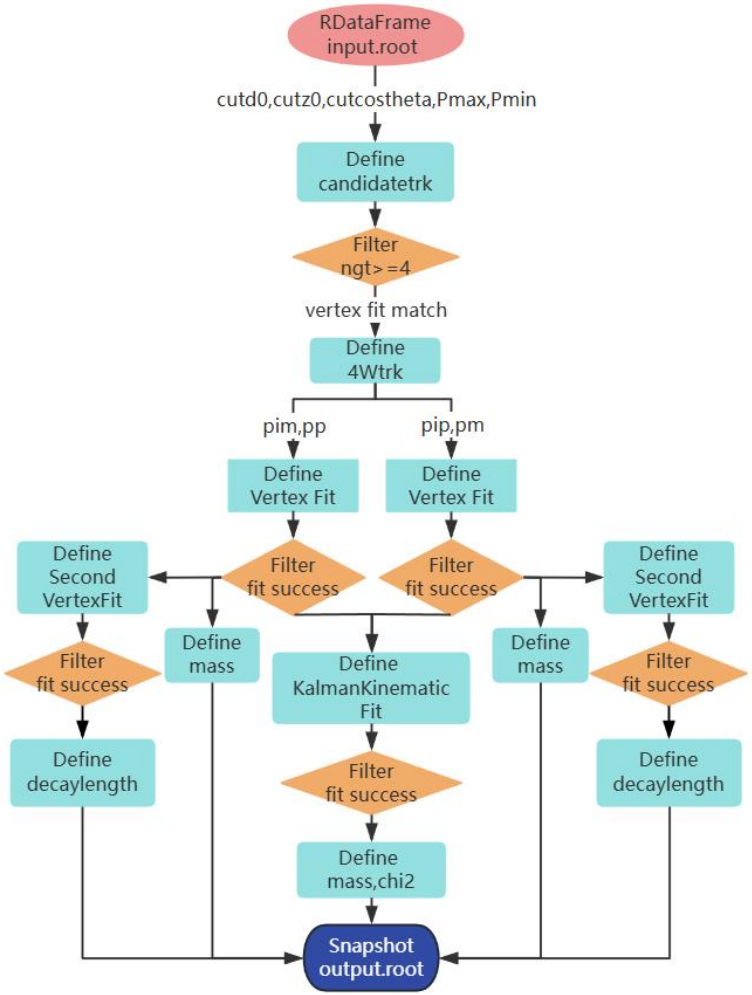
- π^0质量谱



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

核心代码片段

```
# 1. 数据读取与环境初始化
df =ROOT.RDataFrame("events","input.root")
# 2. 径迹参数定义 (底层提取)
df = df.Define("trk_d0","TrackParamHelper::getD0ForHypoPivotIP (...)")
       .Define("trk_z0","TrackParamHelper::getZ0ForHypoPivotIP (...)")
       .Define("trk_mom","TrackParamHelper::getMomForTrack (...)")
# 3. 粒子筛选与物理对象构建
df = df.Define("GoodPtrk","trk_d0<3 && trk_z0<5&&trk_mom>0.5…")
       .Define("TrackPm","TrackerRecTrackCol[GoodPtrk]")
# 4. 顶点拟合
df = df.Define("vfit","VertexFitUtil::get_vfit(...)")
       .Filter("vfit.Fit(0)")
# 5. 运动学拟合
df = df.Define("kfit","KalmanFitUtil::get_kfit(...)")
       .Define("mass_lam","VertexFitUtil::get_massafterkfit(kfit,0,1)")
# 6. 结果输出
df.Snapshot("lambda_result.root", ["decayL","mass_lam",...])
```



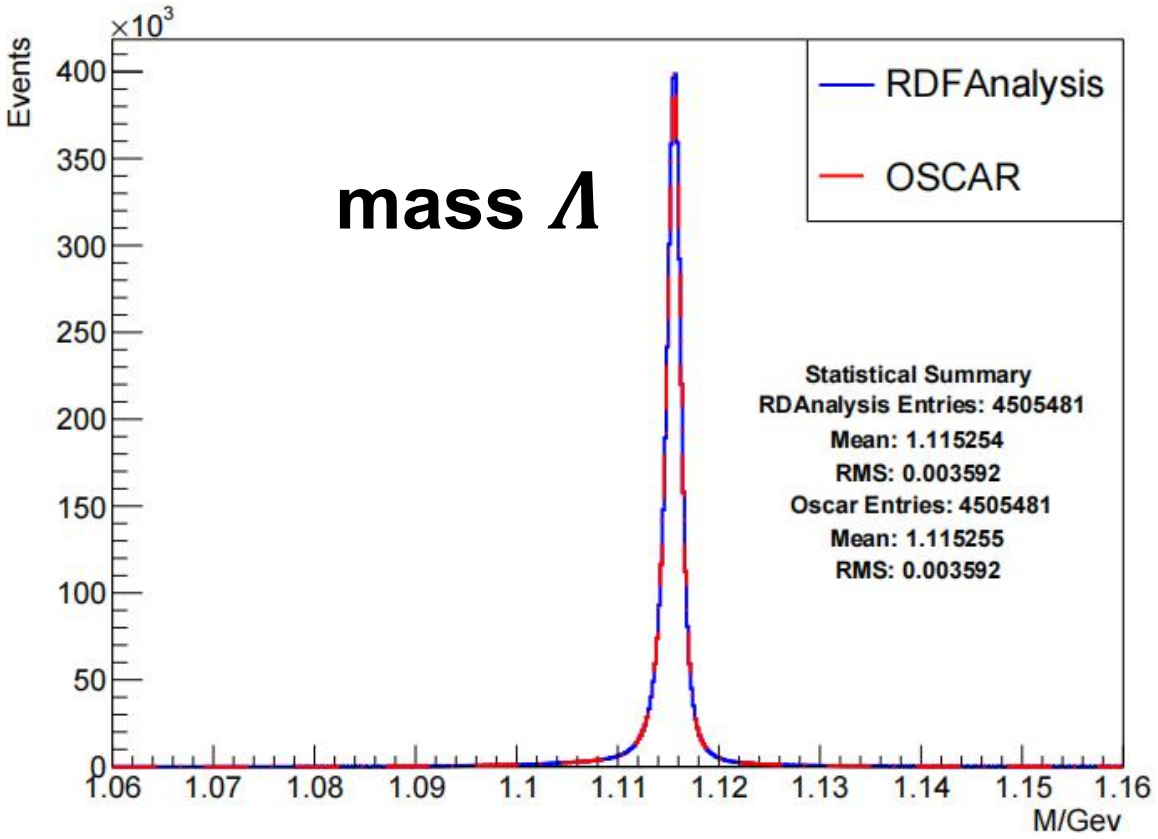
物理分析流程图

https://git.ustc.edu.cn/oscar1/rdfanalysis

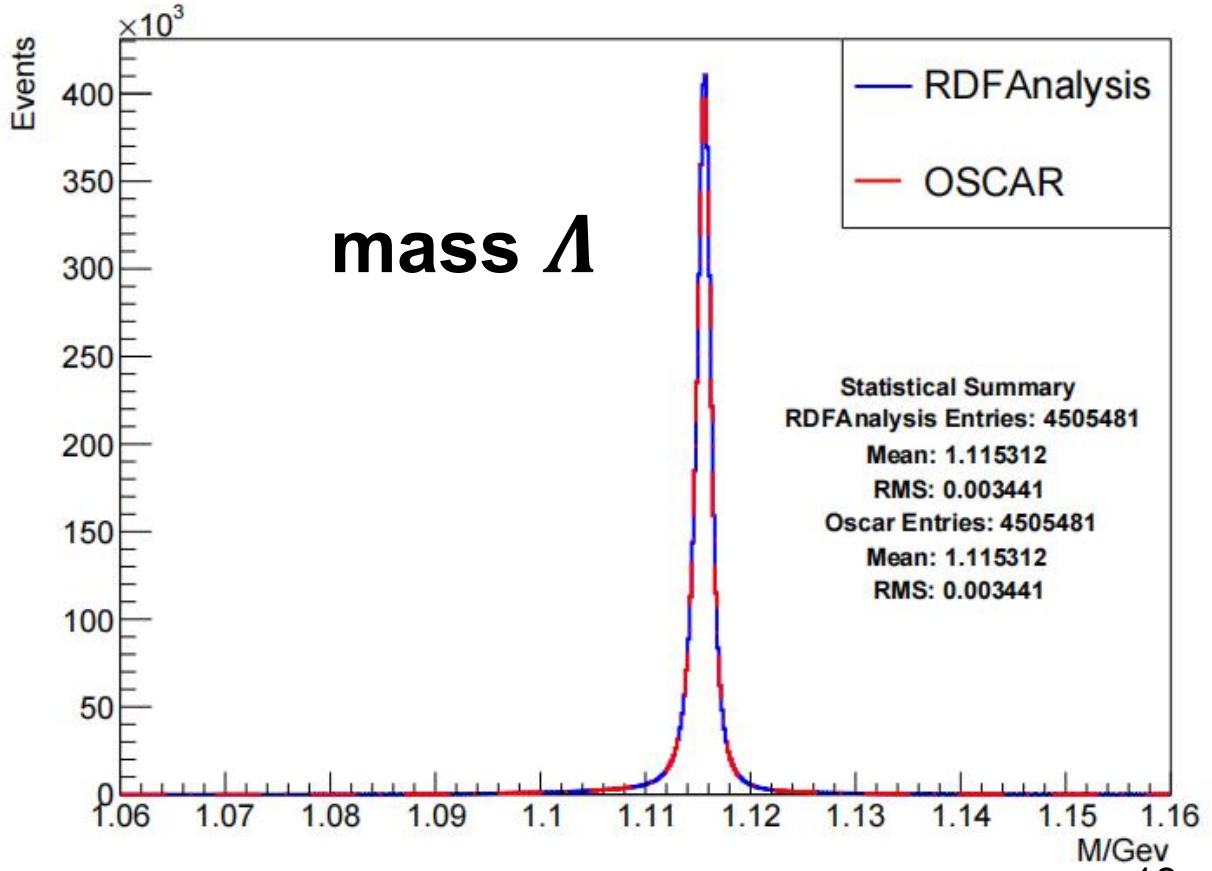
$J/\psi \to \Lambda\bar{\Lambda} \to p\pi^{-}\bar{p}\pi^{+}$

10 million events

顶点拟合



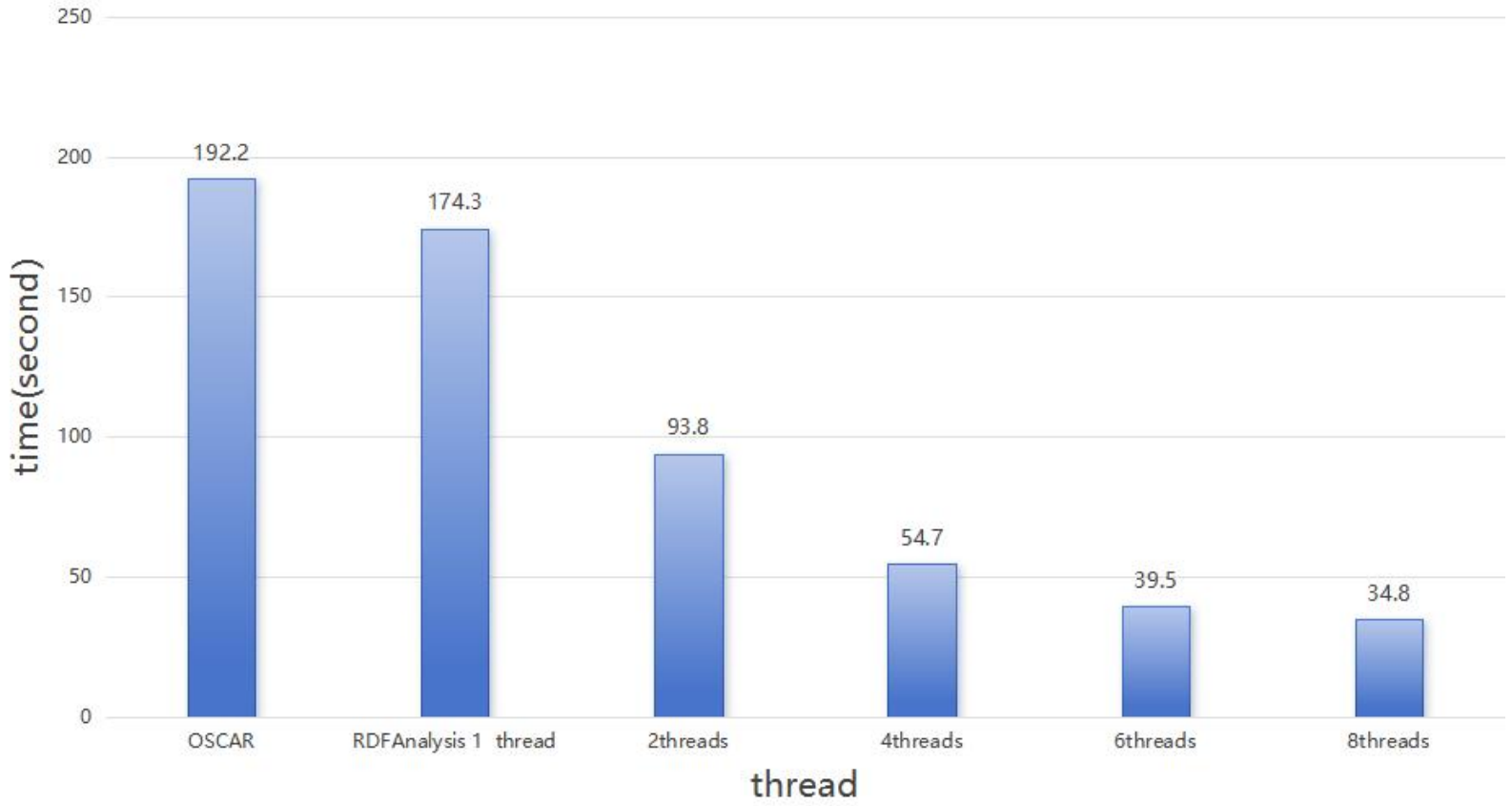
运动学拟合



4 性能测试

性能分析

- 20万 events



- 4线程下，处理速度提升251%
- 受限于数据读取速度和任务调用等必须串行进行的任务，8线程下速度提升不明显

总结

- **框架成功构建**：基于RDataFrame实现STCF并行物理分析框架，集成顶点拟合、运动学拟合、GlobalPID工具
- **物理正确性验证**：$J/\psi \to \rho\pi^0$，$J/\psi \to \Lambda\bar{\Lambda}$ 分析结果与现有OSCAR软件一致，确保了物理分析流程与结论的准确性
- **显著性能提升**：在多线程环境下显著缩短数据分析耗时
- **用户友好的接口设计**：支持声明式编程范式，极大降低了物理学家的使用与开发门槛

展望

- **模型深度适配**：深入对接物理分析专用数据模型，优化数据读写与处理逻辑，进一步释放框架性能潜力
- **工具生态扩展**：持续集成D-tagging等关键物理分析工具，完善框架功能矩阵，覆盖更多研究场景

THANKS!

Reporter:杨莹

Date： 2026.7.2

分析算法示例： $J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$

```
# 创建数据源
df = ROOT.RDataFrame("events", config["input_file"])

# 添加事件索引（从0开始递增）
df_with_index = df.Define("event_index", "rdfentry_")\
                  .Define("n_reconstructed_particles", "ReconstructedParticleCol.size()")\
                  .Define("n_tracks", "TrackerRecTrackCol.size()")

# 1. 径迹选择（使用简化接口）
df1 = build_track_chain(df_with_index, config["hypo_index"])

# 2. PID 分析（使用简化接口）
df2= build_pid_chain("TrackerRecTrackCol",df1)

# 3. 好径迹筛选
df3 = df2.Define("GoodPimtrk",
    f"abs(trk_d0) <= {config['vrCut']} && "
    f"abs(trk_z0) <= {config['vzCut']} && "
    f"trk_charge <= 0 && pi_prob > k_prob && pi_prob > mu_prob")\
    .Define("GoodPiptrk",
    f"abs(trk_d0) <= {config['vrCut']} && "
    f"abs(trk_z0) <= {config['vzCut']} && "
    f"trk_charge > 0 && pi_prob > k_prob && pi_prob > mu_prob")\
    .Define("probpim_pi","pi_prob[GoodPimtrk]")\
    .Define("probpim_k","k_prob[GoodPimtrk]")\
    .Define("Pim", "TrackerRecTrackCol[GoodPimtrk]")\
    .Define("Pip", "TrackerRecTrackCol[GoodPiptrk]")\
    .Filter("Pim.size() == 1 && Pip.size() == 1", "goodtrk")\
    .Redefine("probpim_pi","probpim_pi[0]")\
    .Redefine("probpim_k","probpim_k[0]")

# 4. 获取 WTrackParameter
df4 = df3.Define("pimwtrk", get_wtrack("Pim[0]", config["hypo_index"]))\
         .Define("pipwtrk", get_wtrack("Pip[0]", config["hypo_index"]))

# 5. 中性径迹选择
df5 = df4.Define("nwtrk",
    f"VertexFitUtil::SelectUtil::select_Neutral_Tracks("
    f"RecECALShowerCol, RecExtTrackCol_4, RecExtTrackCol, 2, {config['dangCut']})")\
    .Filter(f"nwtrk.size() >= 2", "neutral tracks")
```

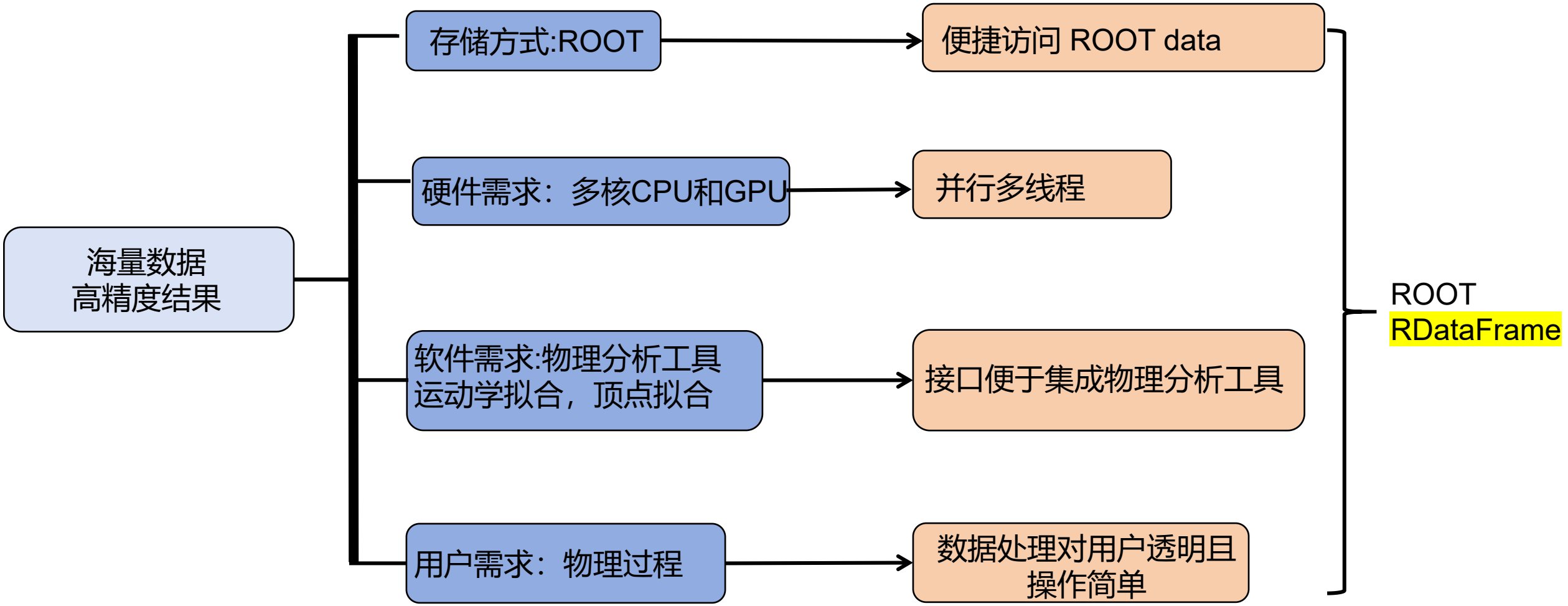
```
# 6. 顶点拟合（使用简化接口）
df6 = df5.Define("vertexfit", get_vfit("pipwtrk", "pimwtrk"))\
         .Filter("vertexfit.Fit(0)", "vfitc")\
         .Define("chi2v","vertexfit.chisq(0)")\
         .Redefine("vertexfit", "VertexFitUtil::update_vfitData(vertexfit)")\
         .Define("kwtrk", "VertexFitUtil::get_forkfit(vertexfit)")\
         .Define("goodnwtrk", "VertexFitUtil::SelectUtil::select_Neuwtrk_forkfit(nwtrk, kwtrk[0], kwtrk[1])"
         .Filter("goodnwtrk.size() == 2", "gn")

# 7. 运动学拟合
df7 = df6.Define("kfit",
    "KinematicFitObject(3.097, kwtrk, goodnwtrk)(2, 2)")\
    .Filter("kfit->Fit()", "kfit")\
    .Define("m_mrho_beforefit", "VertexFitUtil::get_massbeforekfit(kfit, 0, 1)")\
    .Define("m_mrho_afterfit", "VertexFitUtil::get_massafterkfit(kfit, 0, 1)")\
    .Define("m_mpi0_beforefit", "VertexFitUtil::get_massbeforekfit(kfit, 2, 3)")\
    .Define("m_mpi0_afterfit", "VertexFitUtil::get_massafterkfit(kfit, 2, 3)")

# 输出结果（包含事件索引）
df7.Snapshot("events", config["output_file"], [
    "event_index",
    "n_reconstructed_particles",
    "n_tracks",
    "pi_prob",
    "k_prob",
    "mu_prob",
    "trk_d0",
    "trk_z0",
    "trk_pt",
    "m_mpi0_afterfit",
    "m_mrho_afterfit",
    "chi2v",
    "probpim_pi",
    "probpim_k"
])
```

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

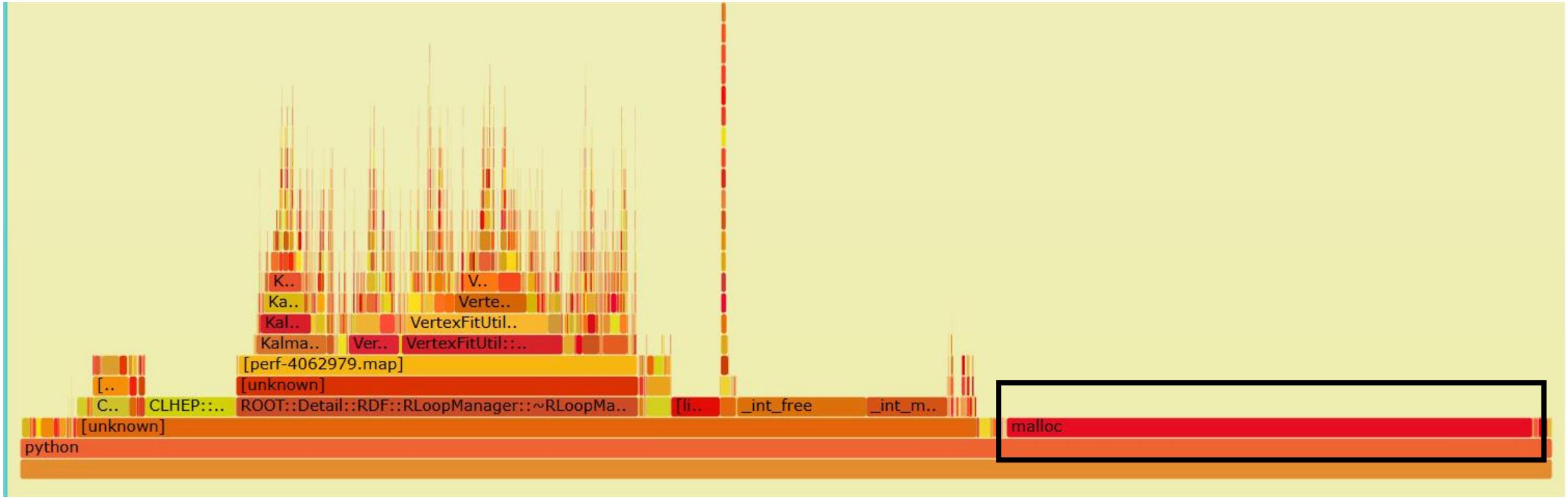
```
df1 = df.Define("event_index", "rdfentry_")\
      .Define("trk_d0", "TrackParamHelper::getD0ForHypoPivotIP(TrackerRecTrackCol, TrackerHypoTrackCol, 2)")\
      .Define("trk_z0", "TrackParamHelper::getZ0ForHypoPivotIP(TrackerRecTrackCol, TrackerHypoTrackCol, 2)")\
      .Define("trk_theta", "TrackParamHelper::getThetaForTrack(TrackerRecTrackCol, TrackerRecTrackCol_1, 2)")\
      .Define("trk_mom", "TrackParamHelper::getMomForTrack(TrackerRecTrackCol, TrackerRecTrackCol_1, 2)")\
      .Define("GoodPtrk",f"abs(trk_d0)<10&&abs(trk_z0)<30&&abs(cos(trk_theta))<0.93 &&abs(trk_mom)>0.5")\
      .Define("GoodPitrk",f"abs(trk_d0)<10&&abs(trk_z0)<30&&abs(cos(trk_theta))<0.93 &&abs(trk_mom)<0.5")\
      .Define("TrackerP","TrackerRecTrackCol[GoodPtrk]")\
      .Define("TrackerPi","TrackerRecTrackCol[GoodPitrk]")\
      .Filter(f"TrackerP.size() >= 2 && TrackerPi.size() >= 2","goodtrk")\
      .Define("Pcharge","TrackParamHelper::getCharge(TrackerP)")\
      .Define("Picharge","TrackParamHelper::getCharge(TrackerPi)")\
      .Define("TrackPm","TrackerP[Pcharge<0]")\
      .Define("TrackPp","TrackerP[Pcharge>0]")\
      .Define("TrackPip","TrackerPi[Picharge>0]")\
      .Define("TrackPim","TrackerPi[Picharge<0]")\
      .Filter("TrackPm.size()>=1 &&TrackPp.size()>=1&&TrackPip.size()>=1&&TrackPim.size()>=1","track")

df_wtrk=df1.Define("wtrks","VertexFitUtil::SelectUtil::select_WTP(TrackPp,TrackPim,TrackerHypoTrackCol,4,2,5,2,8,true)")\
          .Filter("wtrks[0].size()==1&&wtrks[1].size()==1")\
          .Define("Pp","wtrks[0][0]")\
          .Define("Pim","wtrks[1][0]")\
          .Define("Pmpipwtrks","VertexFitUtil::SelectUtil::select_WTP(TrackPm,TrackPip,TrackerHypoTrackCol,4,2,5,2,8,true)")\
          .Filter("Pmpipwtrks[0].size()==1&&Pmpipwtrks[1].size()==1")\
          .Define("Pm","Pmpipwtrks[0][0]")\
          .Define("Pip","Pmpipwtrks[1][0]")

df2=df_wtrk.Define("Vertexfit_p_pi", "VertexFitUtil::get_vfitObject(Pp,Pim)")\
        .Filter("Vertexfit_p_pi.Fit(0)","lamdvfit")\
        .Redefine("Vertexfit_p_pi","VertexFitUtil::update_vfitData(Vertexfit_p_pi)")\
        .Define("wtrkppi", "VertexFitUtil::get_forkfit(Vertexfit_p_pi)")\
        .Define("RDFAm_lambeforefit", "VertexFitUtil::getvfit_Massbefore(Vertexfit_p_pi)")\
        .Define("RDFAm_lamaftervfit", "VertexFitUtil::getvfit_Massafter(Vertexfit_p_pi)")\
        .Define("secondvfit", "VertexFitUtil::get_svfitObject(Vertexfit_p_pi, 5, 2, 8)")\
        .Filter("secondvfit.Fit()","lamdsvfit")\
        .Define("RDFAlam_decaylength", "VertexFitUtil::getsvfit_Decaylength(secondvfit)")\
        .Define("lam_svchi2","secondvfit.chisq()")
```

```
#vertex fit for anti-Lambda
  df3=df2.Define("Vertexfit_pbar_pip",
                 "VertexFitUtil::get_vfitObject(Pm, Pip)")\
         .Filter("Vertexfit_pbar_pip.Fit(0)","antilamd-vfit")\
         .Redefine("Vertexfit_pbar_pip","VertexFitUtil::update_vfitData(Vertexfit_pbar_pip)")\
         .Define("wtrkpbarpip",
                 "VertexFitUtil::get_forkfit(Vertexfit_pbar_pip)")\
         .Define("RDFAm_antilambeforefit",
                 "VertexFitUtil::getvfit_Massbefore(Vertexfit_pbar_pip)")\
         .Define("RDFAm_antilamaftervfit",
                 "VertexFitUtil::getvfit_Massafter(Vertexfit_pbar_pip)")\
         .Define("secondvfit0",
                 "VertexFitUtil::get_svfitObject(Vertexfit_pbar_pip,5, 2, 8)")\
         .Filter("secondvfit0.Fit()","antilamd-svfit")\
         .Define("RDFAantilam_decaylength",
                 "VertexFitUtil::getsvfit_Decaylength(secondvfit0)")\
         .Define("antilam_svchi2","secondvfit0.chisq()")
#kinematic fit
  df4=df3.Define("kfit",f"KalmanKinematicFitObject(3.097,wtrkppi,wtrkpbarpip)(4,0)")\
         .Filter("kfit->Fit()","kfit")\
         .Define("RDFAm_lamafterkfit","VertexFitUtil::get_massafterkfit(kfit,0,1)")\
         .Define("RDFAm_antilamafterkfit","VertexFitUtil::get_massafterkfit(kfit,2,3)")\
         .Define("chi2","VertexFitUtil::get_Chi2afterkfit(kfit)")
  df4.Snapshot("events", f"2thread/RDFA_{i}.root",["lam_svchi2","antilam_svchi2","event_index","RDFAm_lamaftervfit"
```

需求分析


海量数据
高精度结果
存储方式:ROOT
便捷访问 ROOT data
硬件需求：多核CPU和GPU
并行多线程
软件需求:物理分析工具
运动学拟合，顶点拟合
接口便于集成物理分析工具
用户需求：物理过程
数据处理对用户透明且
操作简单
ROOT
RDataFrame

Pref CPU Flame Graph (8 threads)



proportion:34.3% 323e9