山东大学
SHANDONG UNIVERSITY

STCF RDataFrame analysis framework

导师：黄性涛教授　　汇报人：杨莹

日期：2025.7.4

山东大学
SHANDONG UNIVERSITY

目录
Content

1 研究背景与动机

研究背景与动机

山东大学 SHANDONG UNIVERSITY

超级陶粲装置（STCF）

- 质心能量2-7GeV，亮度0.5~1× 10^{35} cm^{-2} s^{-1} (提升50倍)
- 每年产生数据达到几百PB量级，对标准模型的精确检验能力提高1-2量级

计算机由单核向多核发展

- 传统物理分析软件无法充分利用硬件资源，分析速度**受限**
- 并行技术实现多任务并行，**加快**物理分析速度

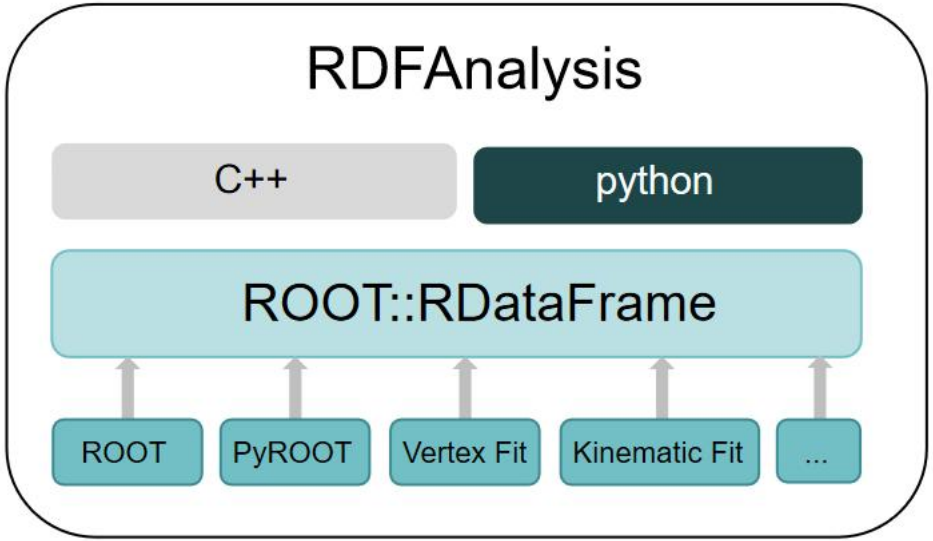
CME (GeV)	Lumi (ab^{-1})	Samples	σ(nb)	No. of Events	Remarks
3.097	1	J/ψ	3400	3.4×10^{12}	
3.670	1	$\tau^+\tau^-$	2.4	2.4×10^{9}	
3.686	1	$\psi(3686)$	640	6.4×10^{11}	
		$\tau^+\tau^-$	2.5	2.5×10^{9}	
		$\psi(3686) \to \tau^+\tau^-$		2.0×10^{9}	
3.770	1	$D^0\bar{D}^0$	3.6	3.6×10^{9}	
		$D^+\bar{D}^-$	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}	
4.180	1	$D_s^{+*}D_s^-$+c.c.	0.90	9.0×10^{8}	
		$D_s^{+*}D_s^-$+c.c.		1.3×10^{8}	Single tag
		$\tau^+\tau^-$	3.6	3.6×10^{9}	
4.230	1	$J/\psi\pi^+\pi^-$	0.085	8.5×10^{7}	
		$\tau^+\tau^-$	3.6	3.6×10^{9}	
		$\gamma X(3872)$			
4.360	1	$\psi(3686)\pi^+\pi^-$	0.058	5.8×10^{7}	
		$\tau^+\tau^-$	3.5	3.5×10^{9}	
4.420	1	$\psi(3686)\pi^+\pi^-$	0.040	4.0×10^{7}	
		$\tau^+\tau^-$	3.5	3.5×10^{9}	
4.630	1	$\psi(3686)\pi^+\pi^-$	0.033	3.3×10^{7}	
		$\Lambda_c\bar{\Lambda}_c$	0.56	5.6×10^{8}	
		$\Lambda_c\bar{\Lambda}_c$		6.4×10^{7}	Single tag
		$\tau^+\tau^-$	3.4	3.4×10^{9}	
4.0–7.0	3	300-point scan with 10 MeV steps, 1 fb^{-1}/point			
> 5	2–7	Several ab^{-1} of high-energy data, details dependent on scan results			

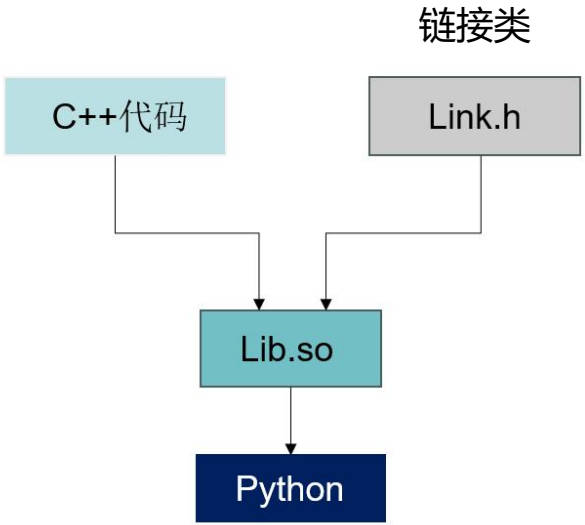
2 框架核心设计

山东大学 SHANDONG UNIVERSITY

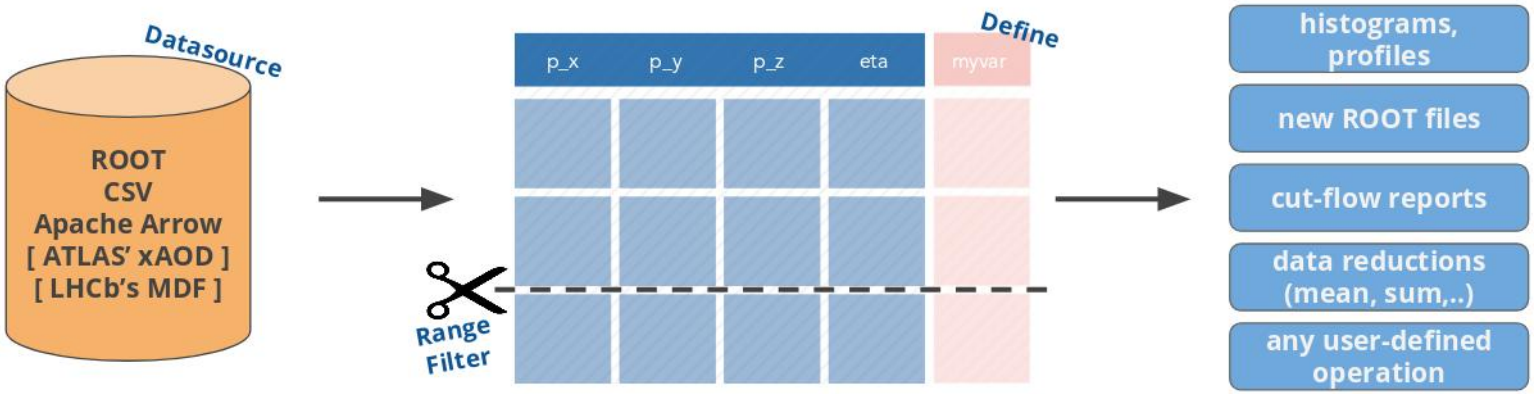
以ROOT：：RDataFrame 为核心构建

- 使用C++与Python混合编程，接口：C++；用户：Python
- 将顶点拟合（拉格朗日乘子法、卡尔曼滤波算法）和运动学拟合（卡尔曼滤波算法）加入框架
- 声明式编程与传统函数编程相结合





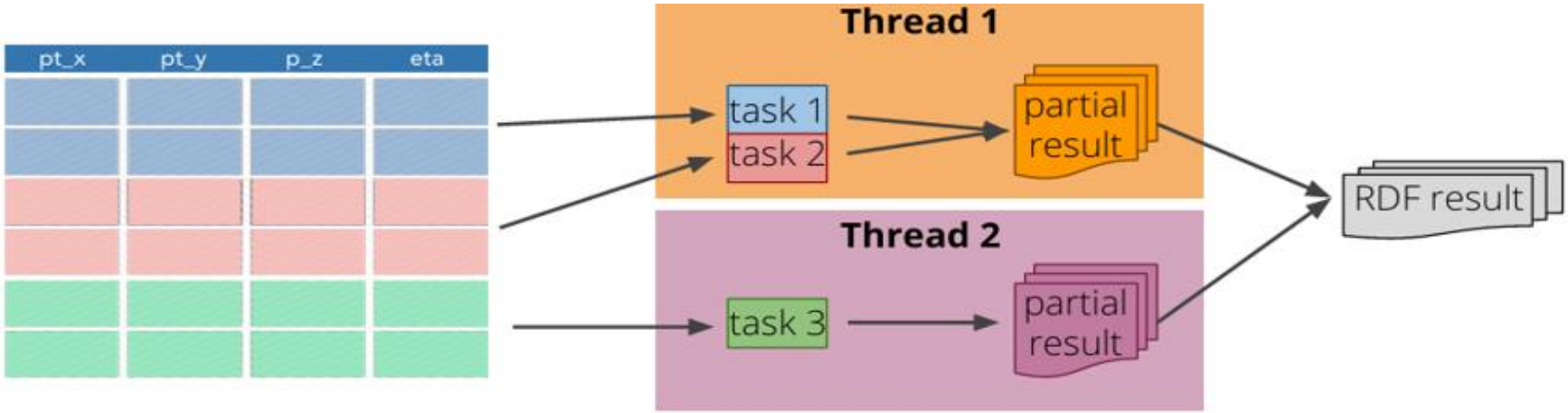
声明式编程与传统函数编程



- 以列的方式处理数据
- 声明式编程
- 传统函数编程

```
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()
```

```
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()
```



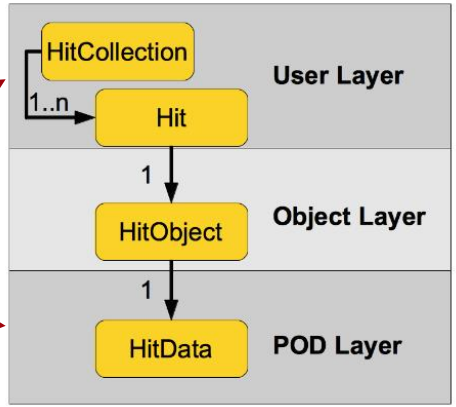
- 对事例生成任务，实现多组数据并行计算
- 只需要一个全局开关开启并行

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

- 具体任务调度由TBB库执行

数据访问模型

- podio生成数据访问:在用户层访问
- RDFAnalysis:在Data层访问



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

接口设计

- 径迹挑选接口

get_2wtrk_byWLpid(
ROOT::VecOps::RVec<GlobalWLPidData>GlobalPidCol, //全局PID数据

ROOT::VecOps::RVec<TrackerRecTrackData> TrackerRecTrackCol, //径迹信息

ROOT::VecOps::RVec<TrackerHypoTrackData> TrackerHypoTrackCol, //假定径迹信息

int PDGID, //挑选粒子径迹的PDGID

int PIDnumber //OSCAR内粒子编号（0-4：e、μ、π、K、p）
)

- 运动学拟合接口

kalmankinematicfit_getObject(WTrackParameter wtrk1,WTrackParameter wtrk2,

WTrackParameter wtrk3,WTrackParameter wtrk4)

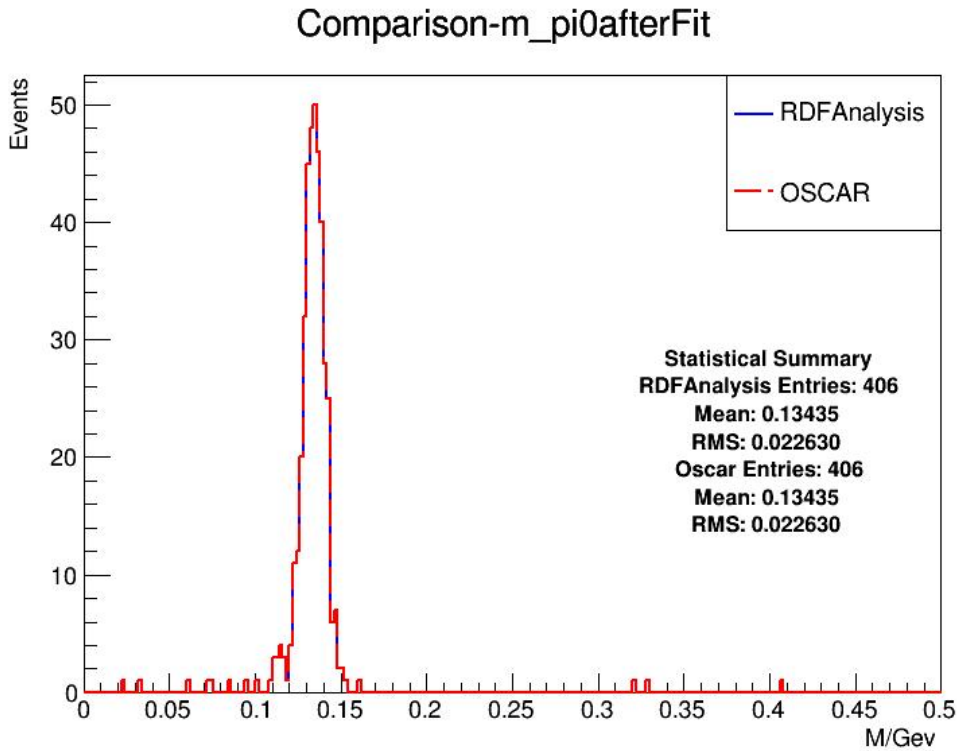
// 输入：4条径迹；输出 KalmanKinematicFit 对象

get_Chi2afterkfit(KalmanKinematicFit kfit)

3 应用举例与验证

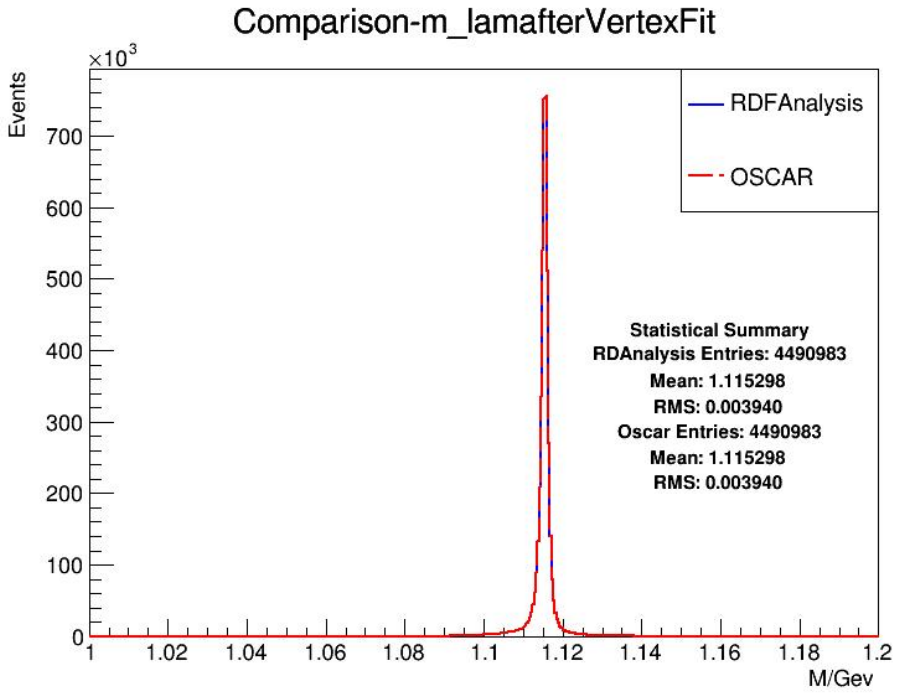
应用举例与验证

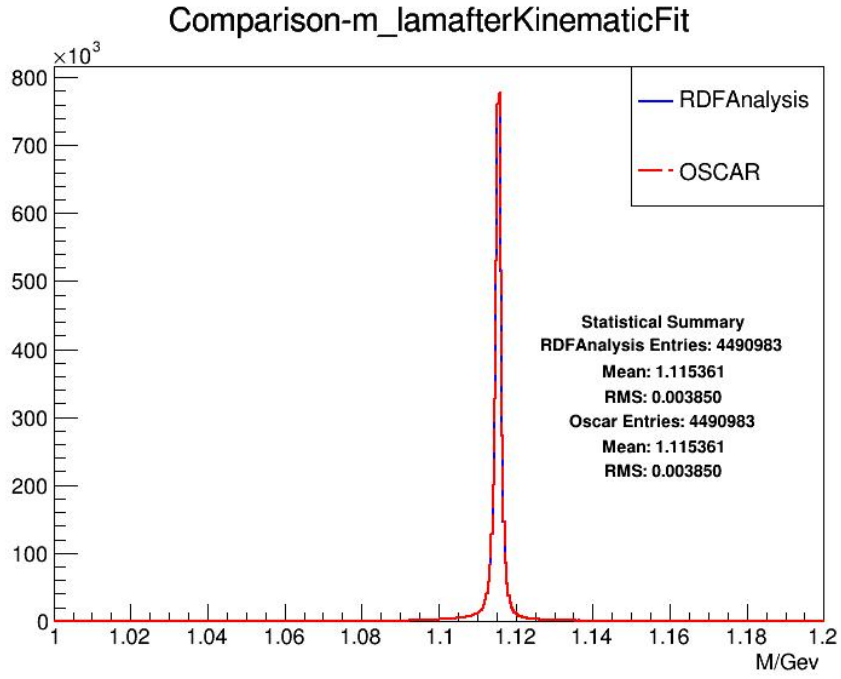
- $J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$



应用举例与验证

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





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

山东大学 SHANDONG UNIVERSITY

径迹挑选

顶点拟合

运动学拟合

```
df1 = df.Define("Goodtrk",
                "VertexFitUtil::SelectUtil::select_iGoodtrk(TrackerRecTrackCol, "
                "TrackerHypoTrackCol, TrackerRecTrackCol_1, 2)")\
        .Filter(f"Goodtrk.size() >= {min_trk}")\
        .Define("ncharge_track",
                "VertexFitUtil::SelectUtil::selectchargetrack(TrackerRecTrackCol, Goodtrk)")\
        .Filter(f"ncharge_track[0] >= {min_pos} && ncharge_track[1] >= {min_neg}")\
        .Define("ppitrk",
                "VertexFitUtil::SelectUtil::select_ppiTrack(TrackerRecTrackCol, TrackerRecTrackCol_1, Goodtrk, 2)")\
        .Filter("ppitrk[0].size() >= 1 && ppitrk[1].size() >= 1 && ppitrk[2].size() >= 1 && ppitrk[3].size() >= 1")\
        .Define("Index",
                f"VertexFitUtil::SelectUtil::select_index(TrackerRecTrackCol, TrackerHypoTrackCol, ppitrk, {particle1_ID}, {particle2_ID}, 5, 2, 8)")\
        .Filter("Index[0] != -1 && Index[1] != -1 && Index[2] != -1 && Index[3] != -1", "cutindex")\
        .Define("Ppwtrk",
                f"VertexFitUtil::SelectUtil::select_wtrk({particle1_mass}, TrackerRecTrackCol, TrackerHypoTrackCol, Index[0], ppitrk[0], {particle1_ID})")\
        .Define("Pmwtrk",
                f"VertexFitUtil::SelectUtil::select_wtrk({particle1_mass}, TrackerRecTrackCol, TrackerHypoTrackCol, Index[1], ppitrk[1], {particle1_ID})")\
        .Define("Pipwtrk",
                f"VertexFitUtil::SelectUtil::select_wtrk({particle2_mass}, TrackerRecTrackCol, TrackerHypoTrackCol, Index[2], ppitrk[2], {particle2_ID})")\
        .Define("Pimwtrk",
                f"VertexFitUtil::SelectUtil::select_wtrk({particle2_mass}, TrackerRecTrackCol, TrackerHypoTrackCol, Index[3], ppitrk[3], {particle2_ID})")\
```

```
# Lambda vertexfit and second vertexfit
df2 = df1.Define("Vertexfit_p_pi", "VertexFitUtil::get_vfitObject(Ppwtrk, Pimwtrk)")\
        .Filter("Vertexfit_p_pi->Fit(0)")\
        .Define("wtrkppi", "VertexFitUtil::get_forkfit(Vertexfit_p_pi)")\
        .Define("wp", "wtrkppi[0]")\
        .Define("wpim", "wtrkppi[1]")\
        .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()")\
        .Define("RDFAlam_decaylength", "VertexFitUtil::getsvfit_Decaylength(secondvfit)")\
```

```
# kinematic fit
df4 = df3.Define("kfit","VertexFitUtil::kalmankinematicfit_getObject(wpim,wp,wpip,wpm)")\
        .Filter("kfit->Fit()")\
        .Define("RDFAm_lamafterkfit","VertexFitUtil::getfirst2trkv_Massafterkfit(kfit)")\
        .Define("RDFAm_antilamafterkfit","VertexFitUtil::getlater2trkv_Massafterkfit(kfit)")\
        .Define("chi2","VertexFitUtil::get_Chi2afterkfit(kfit)")
```

用户界面设置

```
# 1. set input or output
input_file = "/lzufs/user/yangying/RecLam1000w/5000files/reclamphsp262new_2933.root"
output_file = "/lzufs/user/yangying/outputtry/analysis_results.root"
nthread=8
# 2. set
particle_config = {
    'particle1_mass': 0.938272,    # select  particle1 mass
    'particle2_mass': 0.139570,      # select  particle2 mass
    'particle1_ID': 4,    # select  particle1 ID
    'particle2_ID': 2,      # select  particle2 ID
    'min_tracks': 4,            # number of charged track
    'min_positive_charge': 2,    # number of positively charged tracks
    'min_negative_charge': 2     # number of negatively charged tracks
}

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

用户设置：

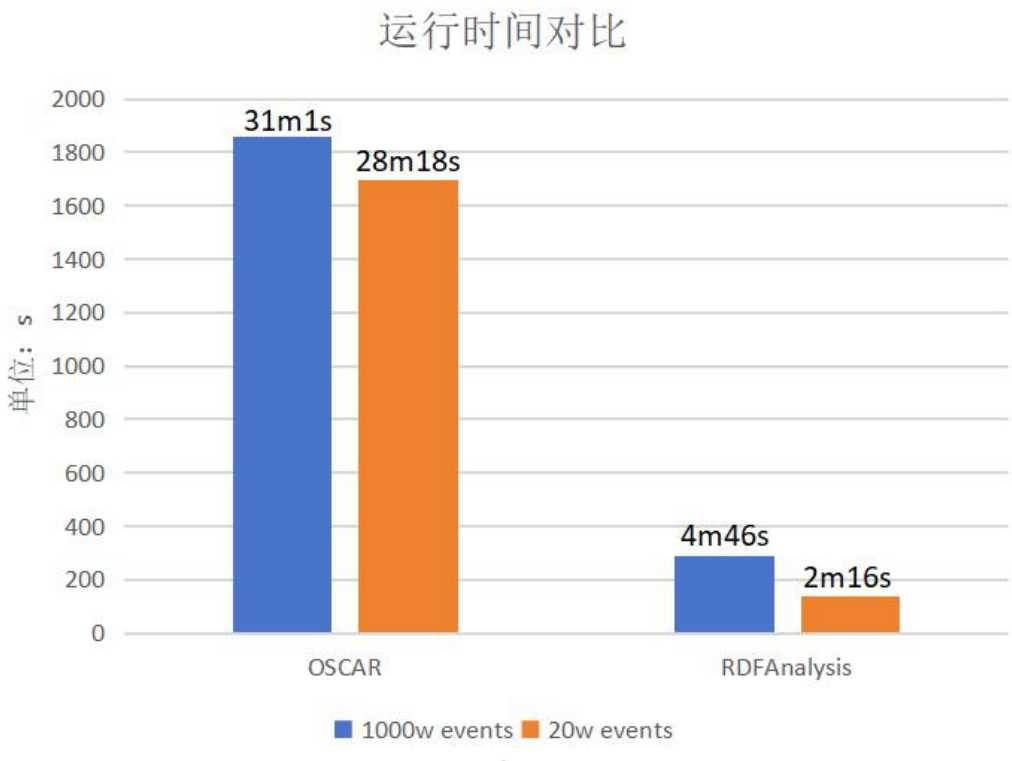
- 输入文件信息
- 输出文件信息
- 线程数
- 挑选带电粒子径迹信息
- 输出参数

4 性能对比

性能对比

- $J/\psi \to \Lambda\overline{\Lambda} \to p\pi^{-}\bar{p}\pi^{+}$
- 传统框架VS并行框架16线程

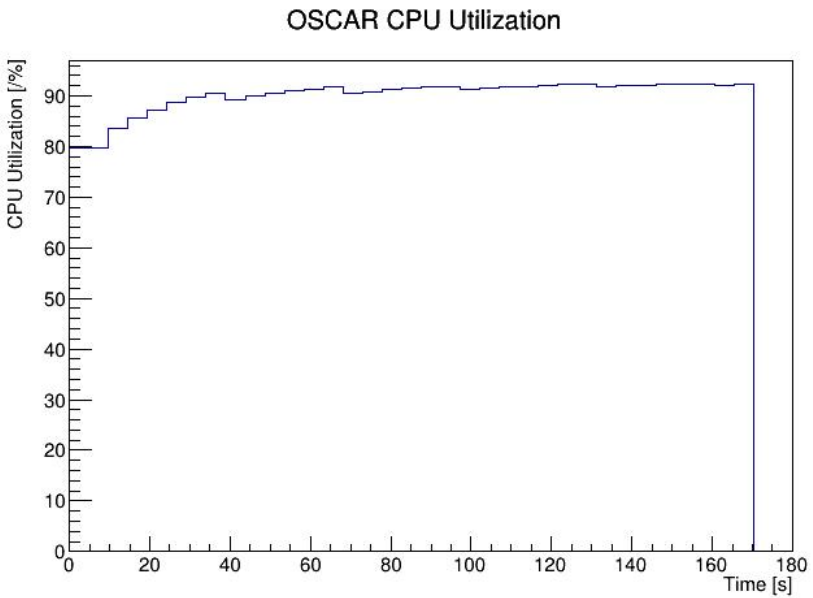
总文件事例数	1000w
总文件大小	169GB
单个文件事例数	20w
单个文件大小	3.38GB

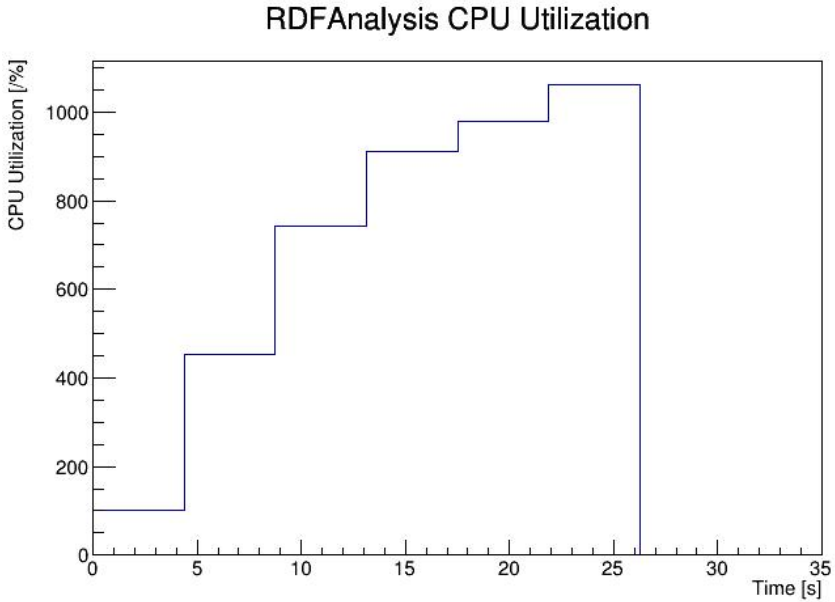


山东大学 SHANDONG UNIVERSITY

CPU利用率比较

- 并行下CPU利用率可高达1000%，充分利用计算资源加快计算速度





总结与展望

STCF RDataFrame analysis framework:

- 使用并行多线程加快物理分析速度，充分利用多核计算机的计算资源
- 使用声明式编程与传统函数编程相结合，提高代码的可读性
- 目前已加入顶点拟合和运动学拟合工具
- $J/\psi \to \rho\pi^0 \to \pi^-\pi^+\gamma\gamma$过程和$J/\psi \to \Lambda\bar{\Lambda} \to p\pi^-\bar{p}\pi^+$过程得到验证

展望：

- 全局顶点拟合等工具将会加入框架
- 更友好的用户界面正在研发，将实现输入物理分析过程就可以输出所需要的数据和图片

感谢!

请各位批评指正

汇报人：杨莹

日期：2025.7.4