基于RDataFrame的并行物理分析框架

方文兴，黄性涛，林韬，李卫东，李刚，李腾，汪圣宜，邹佳恒

大纲

Outline

- 简介
- 设计与开发
- 应用举例
- 性能测试
- 总结与展望

简介

- **现代高能物理实验所产生的分析数据越来越多**
  - 随着高能物理实验的亮度增加和升级，数据的数量和复杂性也在随之增加，硬件技术进步难以应对这些挑战
- **随着CPU向多核方向发展，传统的分析框架面临无法充分利用硬件资源、导致分析效率遇到瓶颈**
- **模拟和重建的并行已经开发并且在框架中实现，但支持并行的分析框架空缺**
- **为了在分析时有效利用计算集群上的所有可用资源，ROOT团队开发了有巨大潜力的 RDataFrame 库**
  - RDataFrame支持隐式多线程，能实现用户透明的并行，充分利用硬件资源，使分析效率大大提高
  - RDataFrame支持声明式编程，能大大简化分析代码
  - RDataFrame提供了一个高级接口，用于分析以 ROOT 格式或其他数据格式存储的数据

Python

```
import ROOT
ROOT.EnableImplicitMT()
# Proceed with the rest of the analysis
```

开启多线程

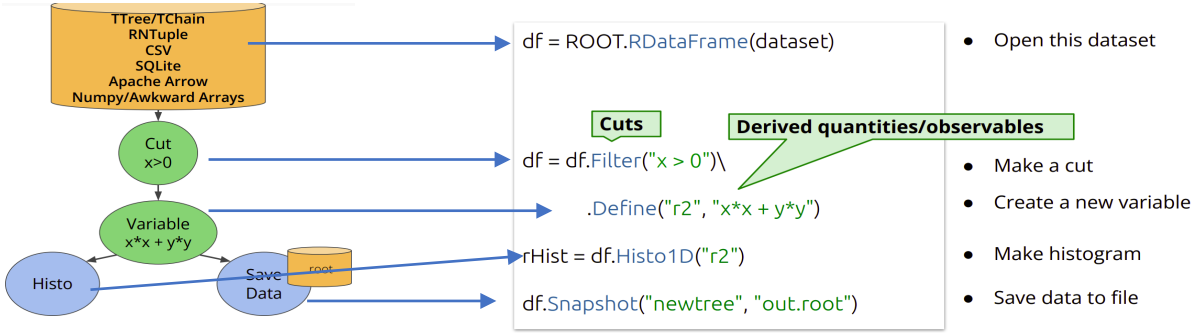
Python

```
h = df.Define("pt", "muon_pt[abs(muon_eta) < 2]").Histo1D("pt")
```

一行代码同时实现cut，定义变量，画图；且代码易读

简介

➢ RDataFrame分析代码



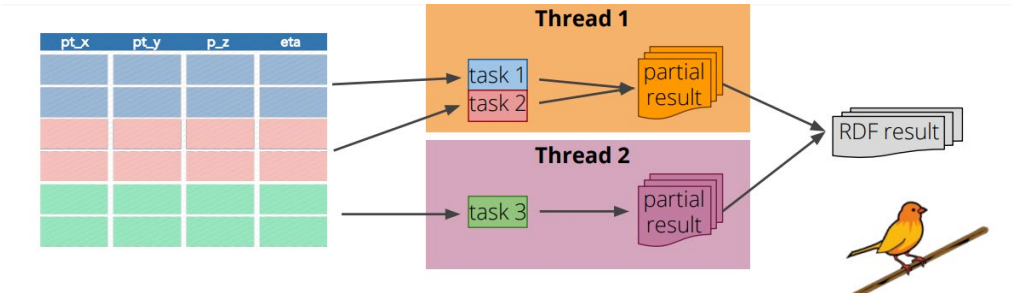
➢ 支持C++&python

```python
h = df.Define("pt", "muon_pt[abs(muon_eta) < 2]").Histo1D("pt")
```

```cpp
RVecD selectPt(RVecD &pt, RVecD &eta) {
  return pt[abs(eta) < 2];
}

auto h = df.Define("pt", selectPt, {"muon_pt", "muon_eta"})
           .Histo1D<RVecD>("pt");
```

➢ 基于任务的并行

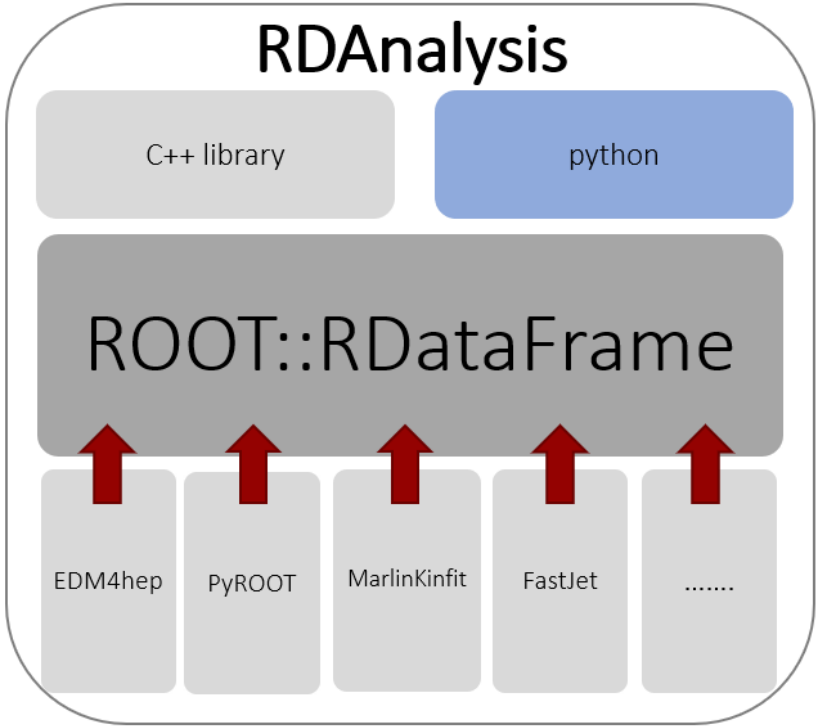
- 每个任务处理某一范围的entries
- Intel TBB作为任务调度器和线程池管理器



简介

- **基于RDataFrame的物理分析工作在国际高能粒子物理实验中得到了广泛应用**
  - 多个国际高能粒子物理实验中已经包含RDataFrame的分析（CMS、ATLAS、ALICE、FCC…）
  - 已有多个基于RDataFrame开发的物理分析框架（bamboo、CROWN、FCCAnalyses…）
  - 未来key4hep计划开发基于RDataFrame的通用物理分析框架K4Analyses
- **为CEPC设计开发基于RDataFrame的物理分析框架是国内首次应用**

设计与开发



- **框架基本结构**
  - 底层外部库：提供物理分析工具
  - C++ library：分析中的数据类型，外部库接口，独立算法等
  - Python：RDataFrame分析通用代码，调用外部C++软件库（面向用户）
- **框架开发内容**
  - 开发读入数据的接口
  - 引入物理分析工具，如运动学拟合程序MarlinKinfit, jet clustering工具FastJet
  - 定义分析常用的数据类型
- **运行环境**
  - CMake:3.16.9
  - ROOT:6.26/8
  - Python:3.10.8
  - gcc : 11.2.0

应用举例

➢ 以e+e- ->H(2jet) mumu为例介绍框架的开发工作

- 取原始数据LCIO并将其转换为ROOT数据
- 剔除muon后将剩余粒子输入Jet clustering程序，重建2Jet
- 将末态2muon2Jet输入运动学拟合程序，对Jet四动量修正

➢ 准备数据

- 利用K4LCIOReader，读入Marlin中产生的LCIO数据并转换成ROOT数据（EDM4hep），供后续分析使用

应用举例

➢ JetClustering

- 基于FastJet(3.4.0)软件，在RDAnalysis中支持kt, Cambridge/Aachen, anti-kt, 及eekt等JetClustering算法

➢ Kinematic fit

- 基于Marlin中的MarlinKinfit软件，在RDAnalysis中支持OPALFitter和NewtonFitter两种运动学拟合算法

➢ 开发

- 编译前cmake中导入外部库FastJet，MarlinKinfit
- 在LinkDef.h中声明分析时所需要使用的类，如clustering_ee_kt，clustering_antikt， LeptonFitObject，OPALFitterGSL
- 在C++库中定义使用JetClustering和运动学拟合的的相关接口及数据类型

应用举例（ e+e- ->H(2jet) mumu ）

- **面向用户的python代码**
  - 由于RDataFrame支持声明式编程，分析代码变得简洁且易读

Step 1
jetclustering.py

```
import ROOT as cepc_ana
import os
cepc_ana.gSystem.Load('./lib/libMarlinKinfit.so')
load=cepc_ana.f()
cepc_ana.ROOT.EnableImplicitMT(12)

df = cepc_ana.RDataFrame("events","~/scratchfs/CEPCSW/0524_eebbMarlin_test.root")

df1=df.Define("cutMuon","ArborPFOs.type!=13&&ArborPFOs.type!=-13")\
.Define("PX1",cepc_ana.get_lep_px("cutMuon"))\
.Define("PY1",cepc_ana.get_lep_py("cutMuon"))\
.Define("PZ1",cepc_ana.get_lep_pz("cutMuon"))\
.Define("E1",cepc_ana.get_lep_e("cutMuon"))\
.Define("pseudo_jets","CEPCAnalyses::JetClusteringUtils::set_pseudoJets(PX1,PY1,PZ1,E1)")\
.Define("CEPCAnalysesJets_ee_genkt", "JetClustering::clustering_ee_kt(2, 2, 0, 0 )(pseudo_jets)")\
.Define("jets_ee_genkt", "CEPCAnalyses::JetClusteringUtils::get_pseudoJets(CEPCAnalysesJets_ee_genkt)")\
.Define("jets_ee_genkt_px","CEPCAnalyses::JetClusteringUtils::get_px(jets_ee_genkt)")\
.Define("jets_ee_genkt_py","CEPCAnalyses::JetClusteringUtils::get_py(jets_ee_genkt)")\
.Define("jets_ee_genkt_pz","CEPCAnalyses::JetClusteringUtils::get_pz(jets_ee_genkt)")\
.Define("jets_ee_genkt_e","CEPCAnalyses::JetClusteringUtils::get_e(jets_ee_genkt)")\
.Define("j2","getJet(jets_ee_genkt_px,jets_ee_genkt_py,jets_ee_genkt_pz,jets_ee_genkt_e)")\
.Snapshot("events","./a1.root",{"j2","ArborPFOs"})
```

打开隐式多线程（12线程）

Cut muon

选择cluster算法并将粒子输入

将cluster得到的jet保存

Step 2
Kinfit.py

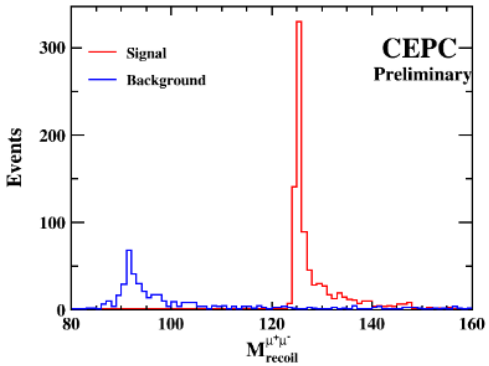
```
df = cepc_ana.RDataFrame("events","./a1.root")

df1=df.Define("Muon",cepc_ana.selLep(13))\
.Define("Muon_px",cepc_ana.get_lep_px("Muon"))\
.Define("Muon_py",cepc_ana.get_lep_py("Muon"))\
.Define("Muon_pz",cepc_ana.get_lep_pz("Muon"))\
.Define("Muon_e",cepc_ana.get_lep_e("Muon"))\
.Filter("Muon_px.size()==2")\
.Define("l1","getLepton(Muon_px,Muon_py,Muon_pz,Muon_e)")\
.Define("higgs_M","Kinfit::fit(j2[0],j2[1],l1[0],l1[1])")\
.Snapshot("events","./recoilR1327hm_4c.root",{"j2","higgs_M","l1"})

df2 = df1.Histo1D(("HiggsRecoM", "after4cfit", 200,0 ,200 ), "higgs_M")
h = cepc_ana.TCanvas()
df2.Draw()
h.SaveAs("higgs.png")
```

输入末态2jet2muon做运动学拟合

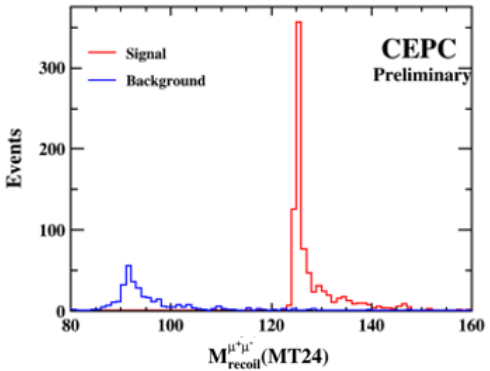
性能测试（ e+e- -> Z(mumu)H ）

- **为测试多线程性能，以e+e- -> Z(mumu)H为例，根据末态2muon重建higgs反冲质量**
  - 与传统框架Marlin对比

- **结果对比**
  - 利用末态2muon的四动量以及质心能量重建Higgs反冲质量，得到与Marlin相同的结果

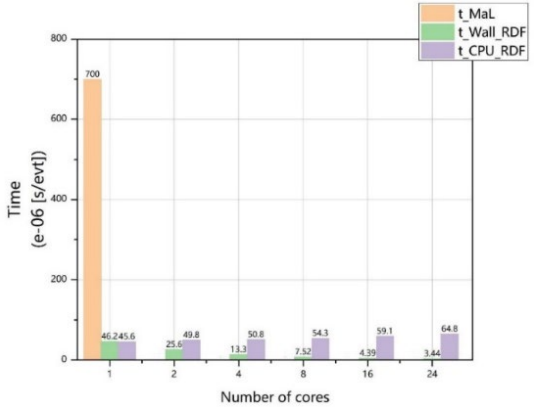


by Marlin



by RDAnalysis

- **性能对比**
  - 多线程下的分析效率较Marlin提升明显

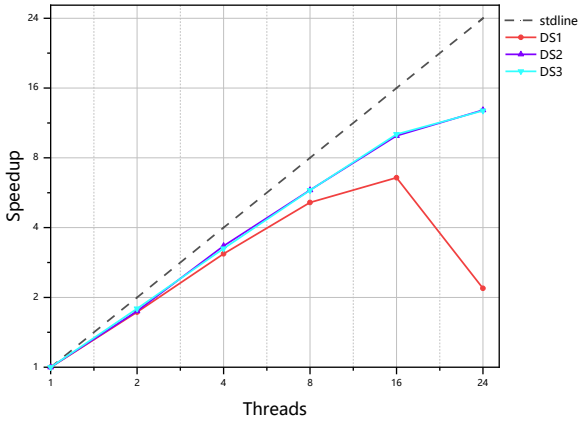


性能测试（e+e- -> Z(mumu)H）

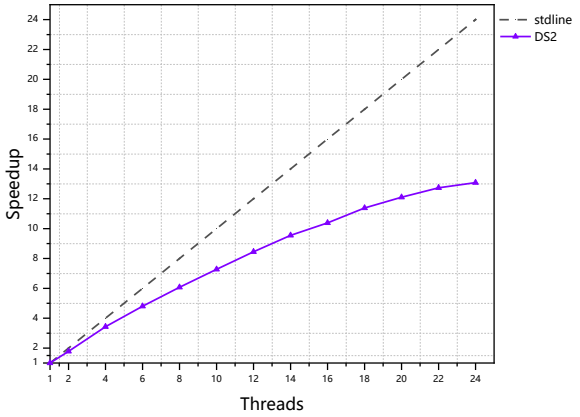
- 以e+e- -> Z(mumu)H为例，根据末态2muon重建higgs反冲质量
  - 扩展性测试

	DS1	DS2	DS3
文件数	20000	348	124
单个文件大小	0.018G	1.03G	2.89G
单个文件Clusters数	1	36	103
总计	358G	358G	358G

测试三种不同数据集对性能的影响



取DS2进一步测试扩展性



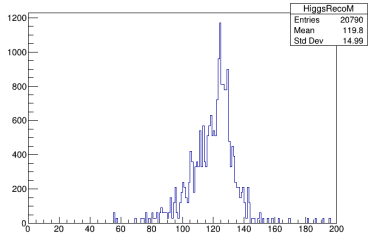
- 在专门的计算节点完成测试(jnws052)
  - Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
- 当单个文件的Clusters数大于所用线程数时，展示出较好的扩展性
- 此物理分析例子对计算资源需求较小，当线程数较多时CPU之间调度的时间占主导

性能测试（ e+e- ->H(2jet) mumu ）

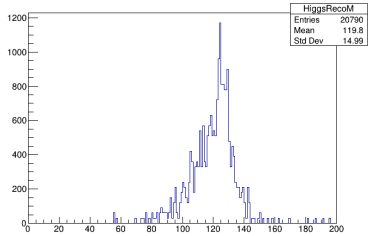
➢ 以e+e- ->H(2jet) mumu为例验证RDAnalysis中jet clustering及运动学拟合程序

➢ Jet clustering性能对比

- 剔除muon后将剩余粒子输入Jet clustering程序，得到2jet重建的higgs质量谱，并与Marlin对比



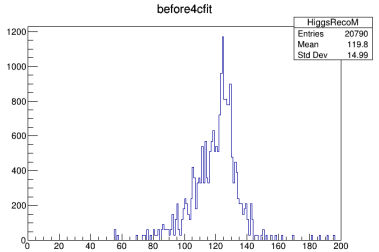
by Marlin

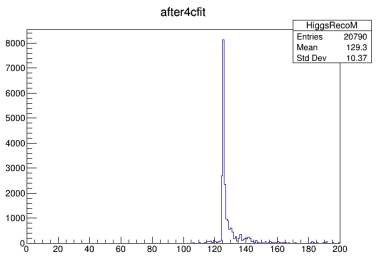


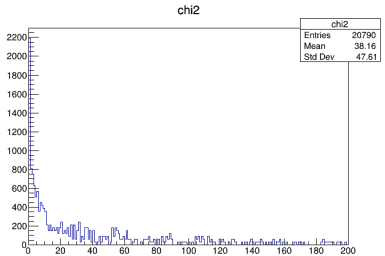
by RDAnalysis

➢ 运动学拟合结果

- 对末态2jet2muon做运动学拟合，得到拟合前后higgs的重建质量谱





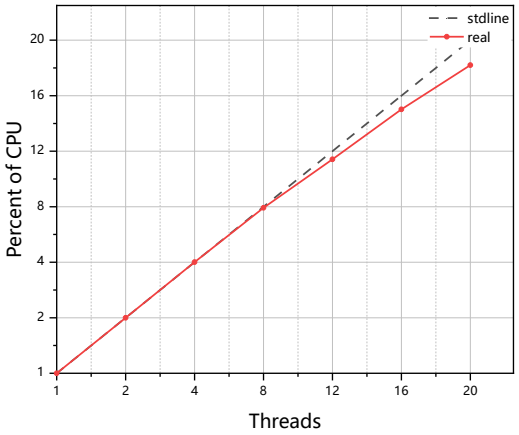


性能测试（ e+e- ->H(2jet) mumu ）

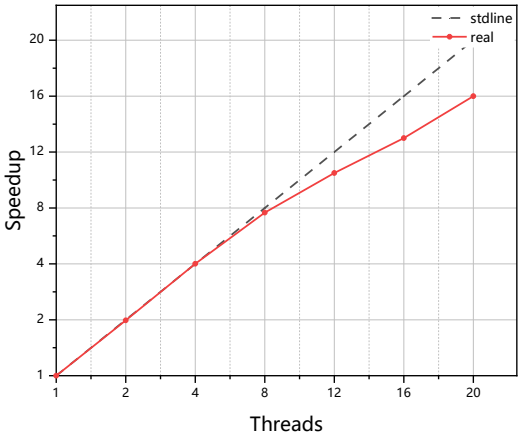
Data Set

文件数	51
单个文件大小	3.1G
单个文件Clusters数	104
总计	156G

CPU利用率随线程数的变化



加速效果随线程数的变化



- CPU利用率随线程数的变化图。随着线程数的增多，CPU利用率的增速稍有下降。
- 加速效果随线程数的变化图。在线程数较多时，由于CPU利用率下降以及CPU之间调度的耗时增加，使得加速效果下降。

总结与展望

工作总结

- **根据广泛调研和前期预研究工作，选择基于RDataFrame开发并行的物理分析框架**
- **为CEPC开发基于RDataFrame的并行分析框架**
  - 初步实现了框架的结构设计
  - 为框架开发了一系列分析所用的数据类型及所需接口，引入运动学拟合及JetClustering软件且验证了有效性
  - 基于RDataFrame的分析框架较传统框架有两大优势，其支持隐式多线程提升了分析效率，同时其声明式编程带来了简单易用的分析代码

工作展望

- **扩充当前分析框架，加入更多物理分析所需算法，如JetTagging, VertexFit**
- **推广基于RDataFrame分析框架在国内高能实验CEPC, STCF, BESIII上的应用**

谢谢大家！

Backup

Loading C++ libraries with dictionaries

For larger analysis frameworks, one may not want to compile the headers each time the Python interpreter is started. One may also want to read or write custom C++/C objects in ROOT files, and use them with RDataFrame or similar tools.

A large analysis framework might further have multiple libraries. In these cases, you generate ROOT dictionaries, and add these to the libraries, which provides ROOT with the necessary information on how to generate Python bindings on the fly. This is what the large LHC experiments do to steer their analysis frameworks from Python.

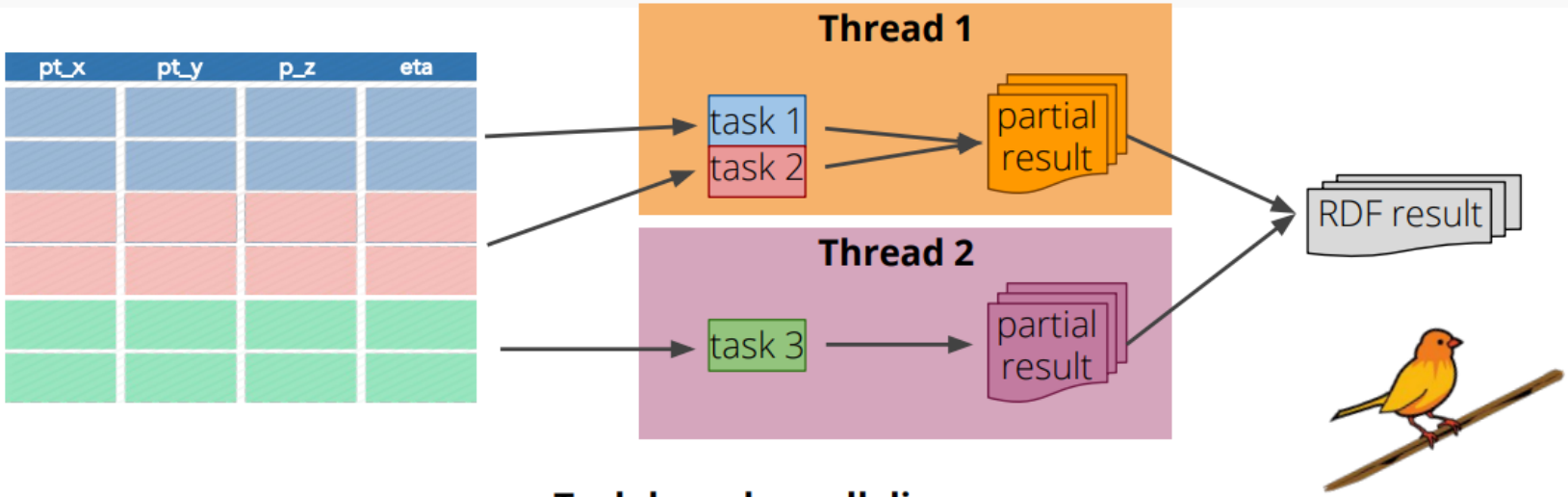
For more information on how to generate ROOT dictionaries, please refer to this section of the manual.

Once the library with dictionaries is available, load it with high-performance C++ in one step:

```
import ROOT
ROOT.gSystem.Load('./libAnalysisLib.so')
```

https://root.cern/manual/io_custom_classes/#using-cmake

RDataFrame's parallelization scheme



Task-based parallelism

- ➔ each task processes a range of entries (thanks to inherent independence of HEP events)
- ➔ **cannot overcommit**, plays well with e.g. experiment frameworks
- ➔ range granularity is the same as TTree compression's to **avoid redundant decompressions**
- ➔ **Intel TBB** is currently ROOT's task scheduler and thread pool manager
- ➔ **RDF parallel *writing*** is also task-based, see G. Amadio, "Writing ROOT Data in Parallel", CHEP 2018

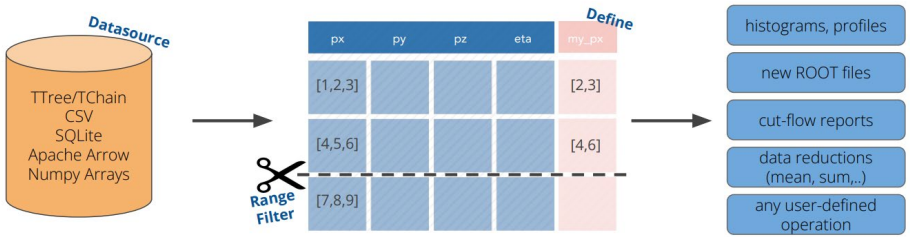
Wide adoption from analysts

- Distributed analysis with RDataFrame in TOTEM (Avati et al., 2019)
- Dark matter sensitivity study (Pani & Polesello, 2018)
- **ATLAS**: prototype xAOD data source DOI 10.5281/zenodo.1303038
- **ALICE**: Apache Arrow support contributed by G. Eulisse
- **FCC** is developing workflows based on RDF
  (C. Helsens, "General status and plans", FCC week 2019, slide 20)
- Building block in INFN analysis facility effort
- many users **"in the wild"**: hundreds of threads tagged #rdataframe on the ROOT forum, about the same as #tree and #hist

RDF as a framework building block

Some examples of analysis software based on RDataFrame

- bamboo (recent talk)
- KIT's CROWN (recent talk)
- W mass analysis framework
- LoopSUSYFrame ATLAS analysis tool
- ("Latinos" CMS framework planning transition to RDF)
- narf (recent talk)
- ...




```
ROOT::EnableImplicitMT();                    Run a multi-thread event loop
ROOT::RDataFrame df(dataset);                on this (ROOT, CSV, ...) dataset
auto df2 = df.Filter("x > 0")                only accept events for which x > 0
             .Define("r2", "x*x + y*y");     define r2 = x² + y²
auto rHist = df2.Histo1D("r2");              plot r2 for events that pass the cut
df2.Snapshot("newtree", "out.root");         write the skimmed data and r2
                                             to a new ROOT file
```