arXiv:2302.09860v1 [hep-ex] 20 Feb 2023

Awkward 与 RDataFrame 互相转换

Ianna Osborne、Jim Pivarski普林斯

顿大学，美国新泽西州普林斯顿，邮编 08544 电
子邮件：ianna.osborne@cern.ch

摘要。Awkward 数组 [1]和 RDataFrame [2] 提供了两种截然不同的大规模计算方式。通过增加二者之间零拷贝转换的能力，用户可以兼得两者之长。这让用户在分析中混合使用不同的软件包和语言时拥有更大的灵活性。在 Awkward Array 第 2 版中，ak.to rdataframe 函数将 Awkward 数组呈现为 RDataFrame 数据源的视图。该视图按需生成，且不会复制数据。列读取器根据视图的运行时类型生成，并传递给一个派生自 ROOT::RDF::RDataSource 的生成数据源。ak.from rdataframe 函数会将选定的列转换为原生 Awkward 数组。本文将讨论利用即时编译技术实现这一功能的细节，并展示如何通过 RDataFrame 的高级接口分析存储在 Awkward 数组中的数据。文中还给出了一些列定义、应用以 C++ 编写的用户定义筛选器，以及将列式数据绘图或提取为 Awkward 数组的示例。最后讨论了当前的限制和未来计划。

1. 引言

Awkward 数组是一个用于处理嵌套和可变大小数据的库，支持任意长度的列表、记录、混合类型和缺失数据，并采用类似 NumPy 的 [3]习惯用法。

在典型的 Awkward 用户分析工作流中，ServiceX [4]和 Uproot [5]提供了对列式数据的访问。数据以 Awkward 数组的形式呈现给用户。用户执行事件筛选，应用系统不确定性，生成直方图，构建统计模型、似然函数，开展统计分析，得到拟合结果并进行诊断。用户可以使用 Python 生态系统中的各种软件包——尤其是 Scikit-HEP 软件包 [6] ——来完成每项任务。

相比之下，RDataFrame 是用于数据分析和处理的声明式并行框架。它通过数据源从列式数据格式中读取数据。它对数据应用转换——即筛选行、定义新列——并生成结果。结果可以是直方图等数据归约、新的 ROOT 文件，或任何其他由用户定义的对象或副作用。

2. Python 生态系统中的 Awkward 数组A

wkward数组和 RDataFrame 是两种截然不同的大规模计算方式。Awkward-RDataFrame 桥接让用户能够更灵活地在分析中混合使用不同的软件包和语言（如有需要，参见图 1）。结合使用 Python 和 C++ 有诸多好处。用户可以在内存中混合使用 Awkward 数组、Numba [8], 和 ROOT C++ 开展分析，无需保存到磁盘，也无需离开当前环境。

一方面，完全在 Python 生态系统中开展分析的用户，可以在原本使用 Awkward 的分析中调用 ROOT C++ 函数或纯 C++，以更快的速度执行。将 Awkward 数组转换为 RDataFrame 的性能开销可以忽略不计，因为这只是一个零拷贝视图；而将 RDataFrame 列表转换为 Awkward 数组的方法将在下文第 4 节中讨论。

另一方面，偏好使用 C++、ROOT 和 RDataFrame 的用户，也可以将数据转换为 Awkward 数组。这种转换为后续操作开辟了许多可能性。

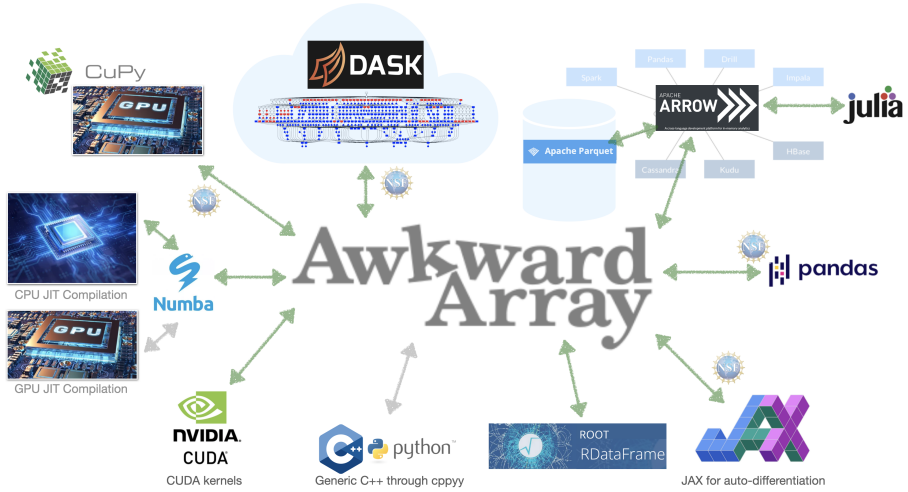

图 1. Awkward Array 与其他项目的互操作性

3. 从 Awkward Array 到 RDataFrame

Awkward 风格的 ak.to rdataframe 函数 [9] 将 Awkward 数组呈现为 RDataFrame 数据源的视图。该视图是一个轻量级的 40 字节 C++ 对象，动态分配在栈上。生成的 RDataSource 通过此视图获取指向原始数组数据的指针。此视图按需生成，不会复制数据——见图 2。只要此函数返回的 RDataFrame 尚未完成执行且当前仍处于运行状态，就可以访问该视图和数组数据。

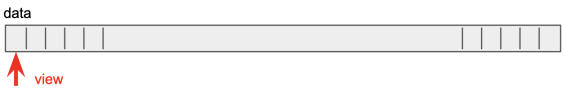

图 2. Awkward Array 视图可以看作一个游标

这项工作利用了一项已有功能：Awkward 数组可以在 Numba 编译函数中迭代，同样使用零拷贝视图。这里复用了 C++ 的 Numba 实现：性能没有差异。列读取器根据视图的运行时类型生成。读取器会传递给一个由 RDataSource 派生的生成的数据源。

ak.to rdataframe 函数接受一个字典作为参数：每个键定义了 RDataFrame 中的一列列名。长度相同的数组作为值赋给字典中的键。数组数据不会被复制，但生成 Awkward RDataSource C++ code 会产生少量开销。

下面的示例介绍了三种典型且各不相同的 Awkward 数组：记录数组、扁平数组和数组列表。将这些数组传递给 RDataFrame，会创建三列不同类型的列。

列“x”将保持其 Awkward 类型，而列“y”将变为整数列，列 z 则是一个变长数组：每个数组都是一个包含双精度值的容器（RVec）。完整教程 [11]（包括对 CMS 开放数据的玩具分析）[12] 已在 PyHEP2022 研讨会上展示。

```
导入 awkward 作为
ak，导入 ROOT

数组_x = ak.Array([
    {"x": [1.1, 1.2, 1.3]} ,
    {"x": [2.1, 2.2]} ,
    {"x": [3.1]} ,
    {"x": [4.1, 4.2, 4.3, 4.4]} ,
    {"x": [5.1]},])
数组_y = ak.Array([1, 2, 3, 4, 5])
数组_z = ak.Array([[1.1], [2.1, 2.3, 2.4], [3.1], [4.1, 4.2, 4.3], [5.1]])

assert len(array_x) == len(array_y) == len(array_z)

df =ak.to_rdataframe({"x": array_x, "y": array_y, "z": array_z})
```

此 ak.to rdataframe 操作不会执行 RDataFrame 事件循环。Awkward RDataSource 会被解释为自定义数据源，其列就是它的数据集：

来自自定义数据源的数据框

属性	值
列总数	4
来自定义的列	1
运行的事件循环	0
处理槽位	1

列	Type	来源
x	awkward::Record_DZ9qK2aXbBA	数据集
y	int64_t	数据集
z	ROOT::VecOps::RVec<双精度浮点数>	数据集

RDataFrame 列的类型是一个字符串，对应于 C++ 数据类型。Awkward 数据类型定义在“awkward” C++ 命名空间中。例如，这里的“x”列包含一个 Awkward 数组，其类型是一个虚构类型：awkward::Record DZ9qK2aXbBA。Awkward 数组采用动态类型，在 C++ 上下文中，类型名称会动态生成，并且名称中包含其内容的哈希值，以确保唯一性。实际上，无需了解该类型。C++ 用户代码应使用占位符类型说明符 auto；所声明变量的类型将根据其初始化器自动推导。

RDataFrame 框架可以对 Awkward 数据执行所有常规操作，只有 Snapshot 操作例外。目前，Awkward 类型列不会保存到 ROOT 文件中。安排操作并不会执行事件循环。例如，以下是对所有数据进行的筛选操作，其中列 "y" 的值大于 2：

df = df.Filter("y % 2 == 0")

筛选后的 Awkward 数组的内部布局——一个 RecordArray 数据：其内容为 NumpyArray——不会被复制，而是会建立索引。由于筛选操作，它被包装在 IndexedArray 中。另外两列数据则会被复制。在 Python 中对 Awkward 数组执行相同操作会产生相同结果：类型相同的数组，但内部布局不同——因为 Awkward 数组是不可变的。

```
array_xyz = ak.Array({"x": array_x, "y": array_y, "z": array_z})
filtered_array = array_xyz[array_xyz["y"] % 2 == 0]
```

4. 从 RDataFrame 转换为 Awkward 数组

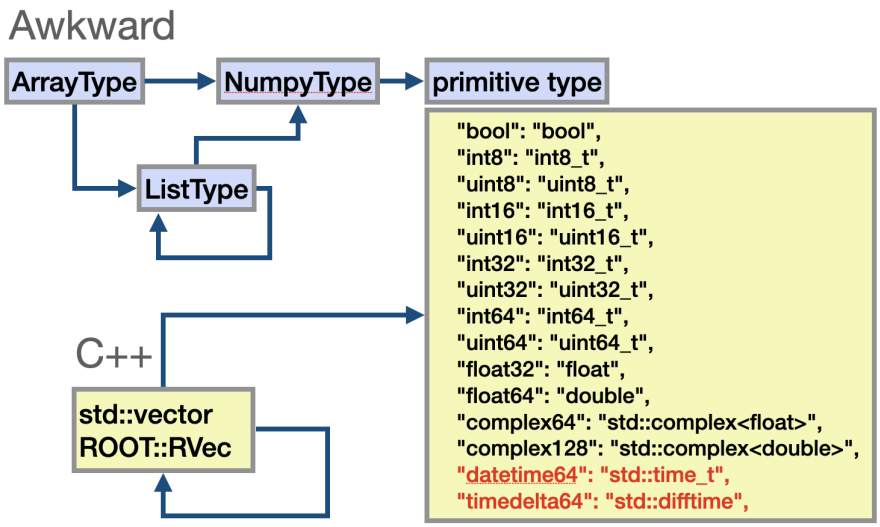

图 3. 支持的 Awkward 数组和 RDataFrame 类型。“基本类型”下方的最后两行显示为红色：它们不受支持。与这些 Awkward 类型对应的 std C 风格日期和时间实用程序库中的 C++ type 尚未完成映射。

ak.from rdataframe 函数 [10]会将选定的列转换为原生 Awkward 数组。该函数接受一个字符串或一个由字符串组成的元组，这些字符串即为 RDataFrame 列名；它可识别以下列数据类型（见图 3.）：

- 基本类型： $integer$、 $float$、 $double$、 $std::complex<double>$ 等。
- 基本类型的列表，以及任意深度嵌套的基本类型列表： $std::vector<double>$、 $RVec<int>$ 等。
- Awkward 类型：由 $awkward::ArrayView$ 或 $awkward::RecordView$ 派生的运行时生成类型。由于 Awkward 数组不可变，因此无需复制——只需将对原始输入数组的引用传递给输出即可。

RDataFrame 事件循环只触发一次，即可获取所有选定的列。

out =ak.from_rdataframe(df, 列=("x", "y", "z",),)

C++ 模板化、仅含头文件的 Awkward-cpp 实现和动态生成的 C++代码均用于提取列类型。这种方法简化了 ROOT 中的 JIT 编译。数组的 Python 字符串描述由 C++ 数据类型构造而成。动态生成的 Python 字符串 Layout 构建器 [13]和生成的函数都用于处理列数据，并通过 Cling [14] 编译。

5. 总结

Awkward 数组和 RDataFrame 提供了两种截然不同的大规模计算方式。通过支持二者之间的转换，用户可以兼得两者的优势。Awkward-RDataFrame 桥接让用户在分析中混用不同的软件包和语言时拥有更大的灵活性。它是 Awkward 2 版的一部分。RootInteractive 项目正在采用这一实现 [15]。用户反馈对于以用户需求为驱动的 Awkward-RDataFrame 后续开发至关重要。

6. 致谢

本工作获得美国国家科学基金会合作协议 OAC-1836650（IRIS-HEP）和美国国家科学基金会合作协议 PHY-2121686（US-CMSLHC 运行）的支持。

参考资料

[1] Pivarski, J., Osborne, I., Ifrim, I., Schreiner, H., Hollands, A., Biswas, A., Das, P., Roy Choudhury, S., Smith, N., Goyal, M.（2018）。Awkward 数组 [计算机软件]。https://doi.org/10.5281/zenodo.4341376[2] RDataFrame：100 线程下的简易并行 ROOT 分析 Danilo Piparo, Philippe Canal, Enrico Guiraud, Xavier Valls Pla, Gerardo Ganis, Guilherme Amadio, Axel Naumann, Enric Tejedor EPJ 网络会议 214 06029 2019 10 1051 201921406029 ( ) DOI：. /epjconf/ [3] Harris, C.R., Millman, K.J., van der Walt, S.J. 等。使用 NumPy 进行数组编程。《自然》585, 357–362（2020 10 1038 41586 020 2649 2 ）。DOI：. /s - - - Galewsky, B., Gardner, R., Gray, L., Neubauer, M., [4] Pivarski, J., Proffitt, M., Vukotic, I., Watts, G., Weinberg, M., ServiceX 分布式缓存列式数据交付服务，EPJ 网络会议 245 04043（2020）DOI: 10 1051 202024504043 . /epjconf/ Pivarski, J., Schreiner, H., Hollands, A., Das, P., Kothari, K., Roy, A., [5] Ling, J., Smith, N., Burr, C., Stark, G.（2017）。Uproot [计算机软件]。https://doi.org/10.5281/zenodo.4340632[6] Scikit-HEP 项目 https://scikit-hep.org[7] Rene Brun, Fons Rademakers, Philippe Canal, Axel Naumann, Olivier Couet, Lorenzo Moneta, Vassil Vassilev, Sergey Linev, Danilo Piparo, Gerardo GANIS, Bertrand Bellenot, Enrico Guiraud, Guilherme Amadio, wverkerke, Pere Mato, TimurP, Matevˇz Tadel, wlav, Enric Tejedor, . . . Raphael Isemann.（2019）。root-project/root：v6.18/02（v6-18-02）。Zenodo。https://doi.org/10.5281/zenodo.3895860[8] Siu Kwan Lam, stuartarchibald, Antoine Pitrou, Mark Florisson, Stan Seibert, Graham Markall, esc, Todd A. Anderson, rjenc29, Guilherme Leobas, luk-f-a, Michael Collison, Jay Bourque, Aaron Meurer, Travis E. Oliphant, Nick Riasanovsky, Kaustubh, Michael Wang, densmirn, . . . James Bourbeau.（2022）。numba/numba：版本 0.56.4（0.56.4）。Zenodo。https://doi.org/10.5281/zenodo.7289231[9] https://awkward-array.org/doc/main/reference/generated/ak.to_rdataframe.html[10] https://awkward-array.org/doc/main/reference/generated/ak.from_rdataframe.html [11] Osborne, I., Pivarski, J.,（2022 年 9 月 15 日）。Awkward RDataFrame 教程。PyHEP 2022（线上）研讨会。Zenodo。https://doi.org/10.5281/zenodo.7081586[12] CERN 开放数据门户 DOI:10.7483/OPENDATA.CMS.LVG5.QT81（CMS 数据）[13] Goyal, M., Osborne, I., Pivarski, J. Python 与 C++的 Awkward 世界，本文集 [14] Cling [在线]。https://root.cern.ch/cling[15] ROOTInteractive https://github.com/miranov25/RootInteractive