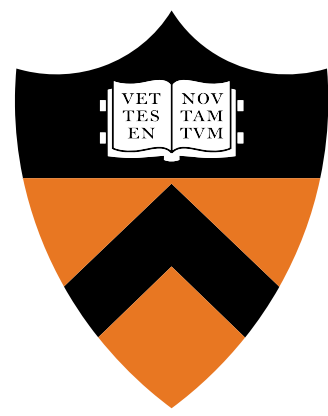


Current Status of AwkwardArray.jl:

Advancing Interoperability in High-Energy Physics Workflows

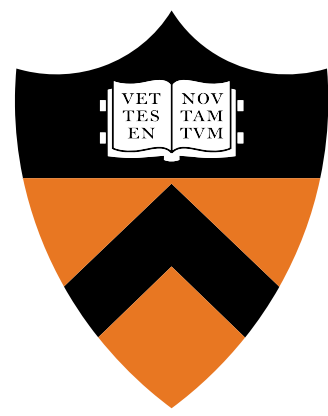
Ianna Osborne, Princeton University



HEP Software Migration to Julia

Development strategies



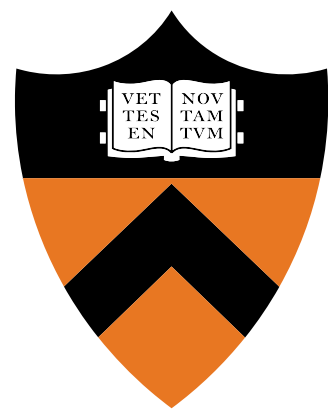


HEP Software Migration to Julia



Development strategies

- Wrapping mature C++ libraries to make them accessible from Julia — for example, Geant4.jl provides a Julia interface to the widely used Geant4 particle transportation toolkit, and ROOT.jl offers bindings to the ROOT data analysis framework.

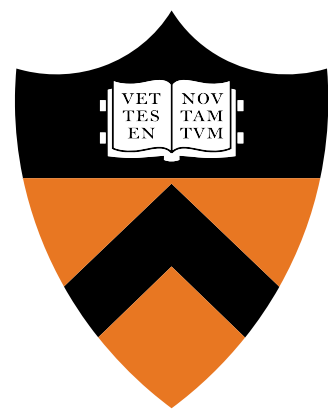


HEP Software Migration to Julia



Development strategies

- Wrapping mature C++ libraries to make them accessible from Julia — for example, Geant4.jl provides a Julia interface to the widely used Geant4 particle transportation toolkit, and ROOT.jl offers bindings to the ROOT data analysis framework.
- Opt for complete reimplementations of existing tools in native Julia, as demonstrated by BAT.jl for Bayesian analysis and UnROOT.jl for reading ROOT files without relying on the C++ libraries.

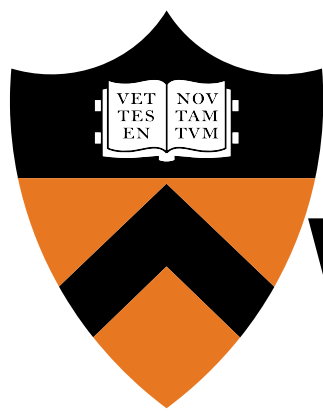


HEP Software Migration to Julia



Development strategies

- Wrapping mature C++ libraries to make them accessible from Julia — for example, Geant4.jl provides a Julia interface to the widely used Geant4 particle transportation toolkit, and ROOT.jl offers bindings to the ROOT data analysis framework.
- Opt for complete reimplementations of existing tools in native Julia, as demonstrated by BAT.jl for Bayesian analysis and UnROOT.jl for reading ROOT files without relying on the C++ libraries.
- We focus on integrating Julia with the Python-based Awkward Array library via its dedicated Julia backend, AwkwardArray.jl, enabling efficient access to nested, irregular data structures commonly used in HEP.



What is Awkward Array

Python library

Example of an “awkward” array

```
array = ak.Array([
    [{"x": 1.1, "y": [1]}, {"x": 2.2, "y": [1, 2]}, {"x": 3.3, "y": [1, 2, 3]}],
    [{"x": 4.4, "y": [1, 2, 3, 4]}, {"x": 5.5, "y": [1, 2, 3, 4, 5]}]
```

Record structures with differently typed fields

Array of variable-length lists (“ragged” or “jagged” arrays)

Nested variable-length lists

Missing data

Heterogenous data (union/variant types)

NumPy-like expression

```
output = np.square(array["y", ..., 1:])
```

Result

```
output.to_list()
[
  [[], [4], [4, 9]],
  [],
  [[4, 9, 16], [4, 9, 16, 25]]
]
```

(for a similar calculation 10 million times larger, single threaded)

1.5 seconds
2.1 GB of memory

Equivalent Python

```
output = []
for sublist in python_objects:
    tmp1 = []
    for record in sublist:
        tmp2 = []
        for number in record["y"][1:]:
            tmp2.append(np.square(number))
        tmp1.append(tmp2)
    output.append(tmp1)
```

140 seconds
22 GB of memory

- An array library for **nested, variable-sized data**, including arbitrary-length lists, records, mixed types, and missing data, using **NumPy-like idioms**.

- Arrays are **dynamically typed**, but operations on them are **compiled and fast**.

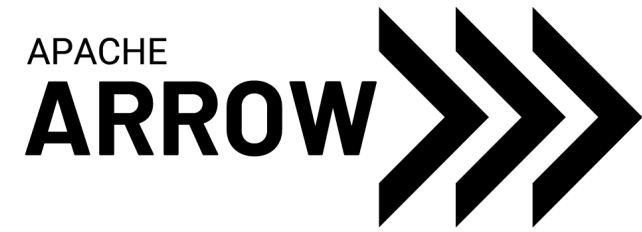
- Coincides with NumPy when arrays are regular; generalizes when they're not.



Awkward Array Library Integrations



Awkward Array is a part of larger scientific Python ecosystem. It must work well with other libraries.



Lossless, bidirectional, zero-copy conversions between Awkward Array, Apache Arrow, Feather, and Parquet.



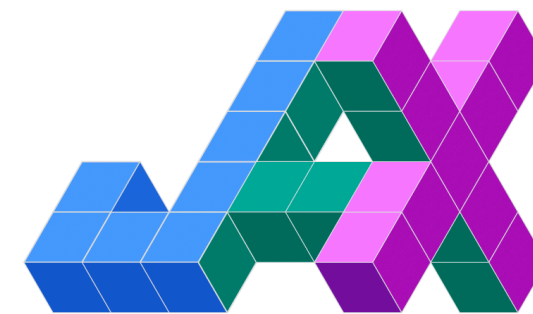
JIT-compiled as a C++ iterator in ROOT's RDataFrame workflow.



Awkward Arrays on GPUs, backed by CuPy.



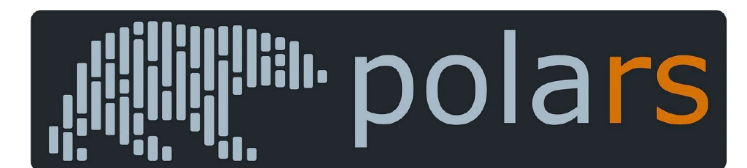
JIT-compile Awkward Arrays in C++ with cppy. Header-only library to build Awkward Arrays in C++ and transfer them to Python.



Auto-differentiate through expressions involving Awkward Arrays, using JAX



Efficiently iterate over or build Awkward Arrays in code JIT-compiled with Numba, on CPUs and GPUs



RAPIDS

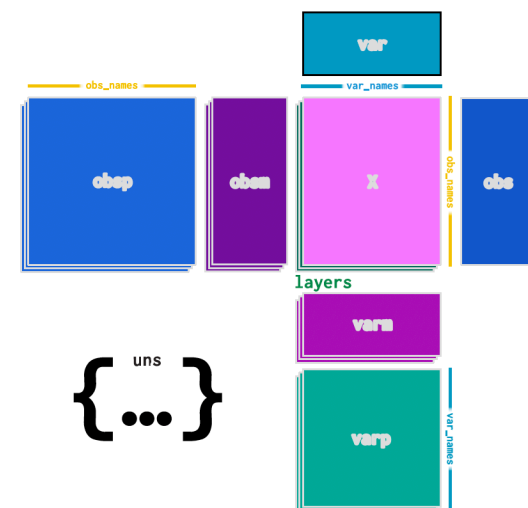
Akimbo: work with Awkward Arrays as DataFrame columns

[ragged]

Manipulating ragged arrays as though they were NumPy or CuPy arrays, following the [Array API specification](#).



Bidirectional conversion between Awkward Array and TensorFlow RaggedTensor.



Integrated into AnnData for single-cell genomics.



Read any Kaitai Struct format into Awkward Arrays.



Store Awkward Arrays in the Tiled database and slice them in the cloud.

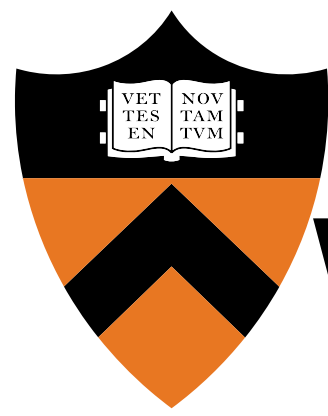


Exchange data with the Julia language through a reimplementation of the Awkward Array memory layouts.



dask-awkward is a high-level collection type for Dask, alongside Array and DataFrame.

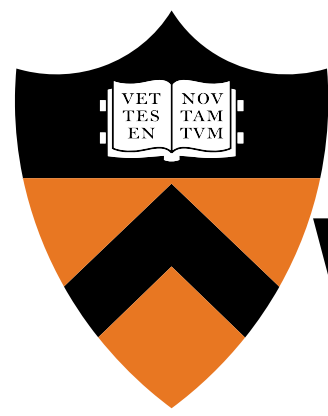




What is Awkward Array.jl

Julia backend

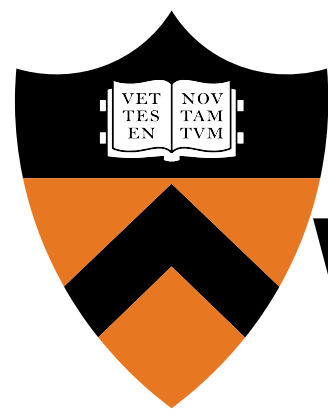
- Awkward Array is a library for handling jagged HEP data in the Python ecosystem.



What is Awkward Array.jl

Julia backend

- Awkward Array is a library for handling jagged HEP data in the Python ecosystem.
- AwkwardArray.jl - a dedicated backend - allows accessing this data from Julia.

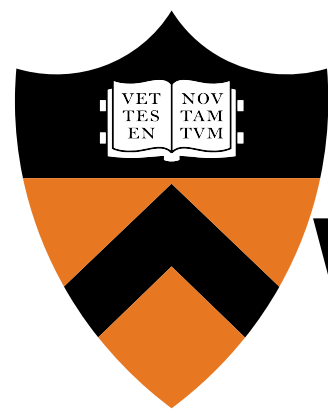


What is Awkward Array.jl

Julia backend



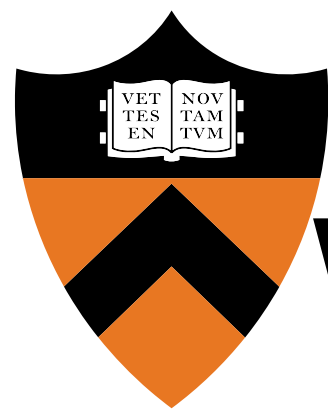
- Awkward Array is a library for handling jagged HEP data in the Python ecosystem.
- AwkwardArray.jl - a dedicated backend - allows accessing this data from Julia.
- To bridge the Python and Julia ecosystems, we use PythonCall.jl, a package that provides bi-directional interoperability between the two languages.



What is Awkward Array.jl

Julia backend

- Awkward Array is a library for handling jagged HEP data in the Python ecosystem.
- AwkwardArray.jl - a dedicated backend - allows accessing this data from Julia.
- To bridge the Python and Julia ecosystems, we use PythonCall.jl, a package that provides bi-directional interoperability between the two languages.
- This enables us to offload computationally heavy operations to Julia — following a pattern familiar from other JIT compilation tools in Python, such as Numba.



What is Awkward Array.jl

Julia backend

- CHEP 2024: evaluated integration by comparing performance and usability using Julia-native types like Vector of Vectors, which provide similar capabilities for representing nested, irregular data.

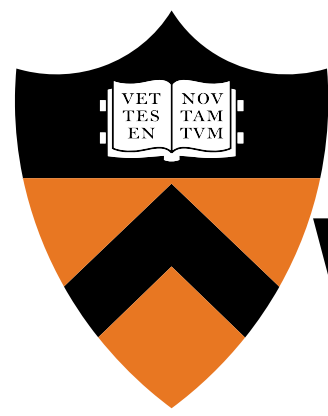


What is Awkward Array.jl

Julia backend



- CHEP 2024: evaluated integration by comparing performance and usability using Julia-native types like Vector of Vectors, which provide similar capabilities for representing nested, irregular data.
- no significant overhead when combining the two ecosystems.

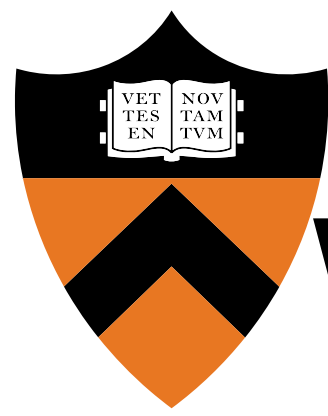


What is Awkward Array.jl

Julia backend



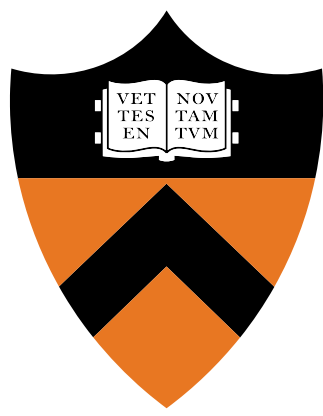
- CHEP 2024: evaluated integration by comparing performance and usability using Julia-native types like Vector of Vectors, which provide similar capabilities for representing nested, irregular data.
- no significant overhead when combining the two ecosystems.
- user interest in exploring Julia kernels for Awkward Array: combinatorics.



What is Awkward Array.jl

Julia backend

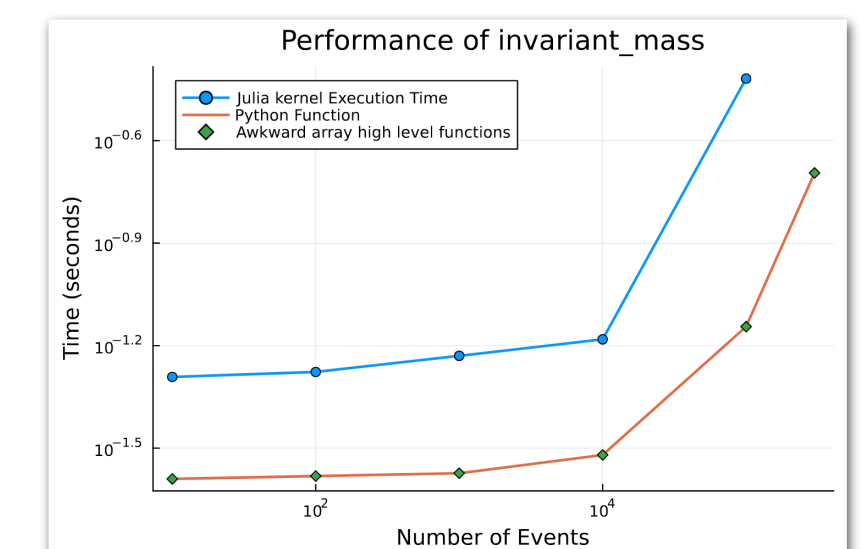
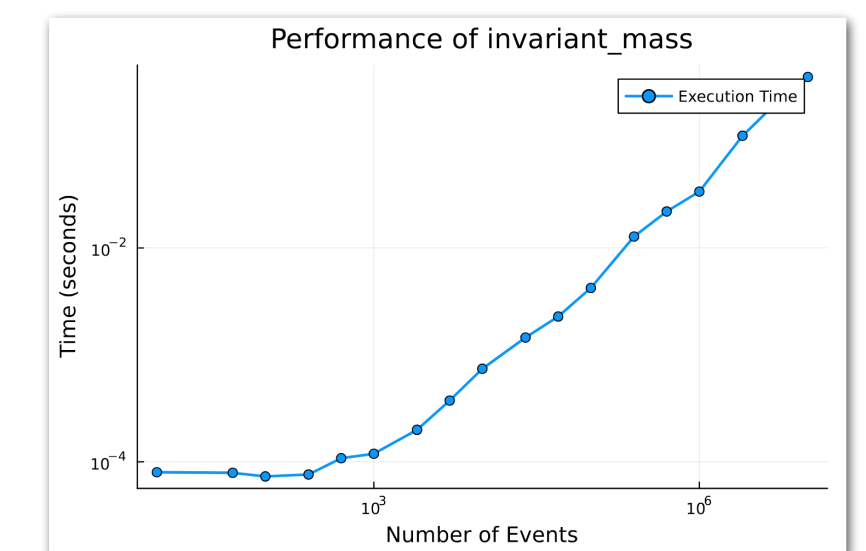
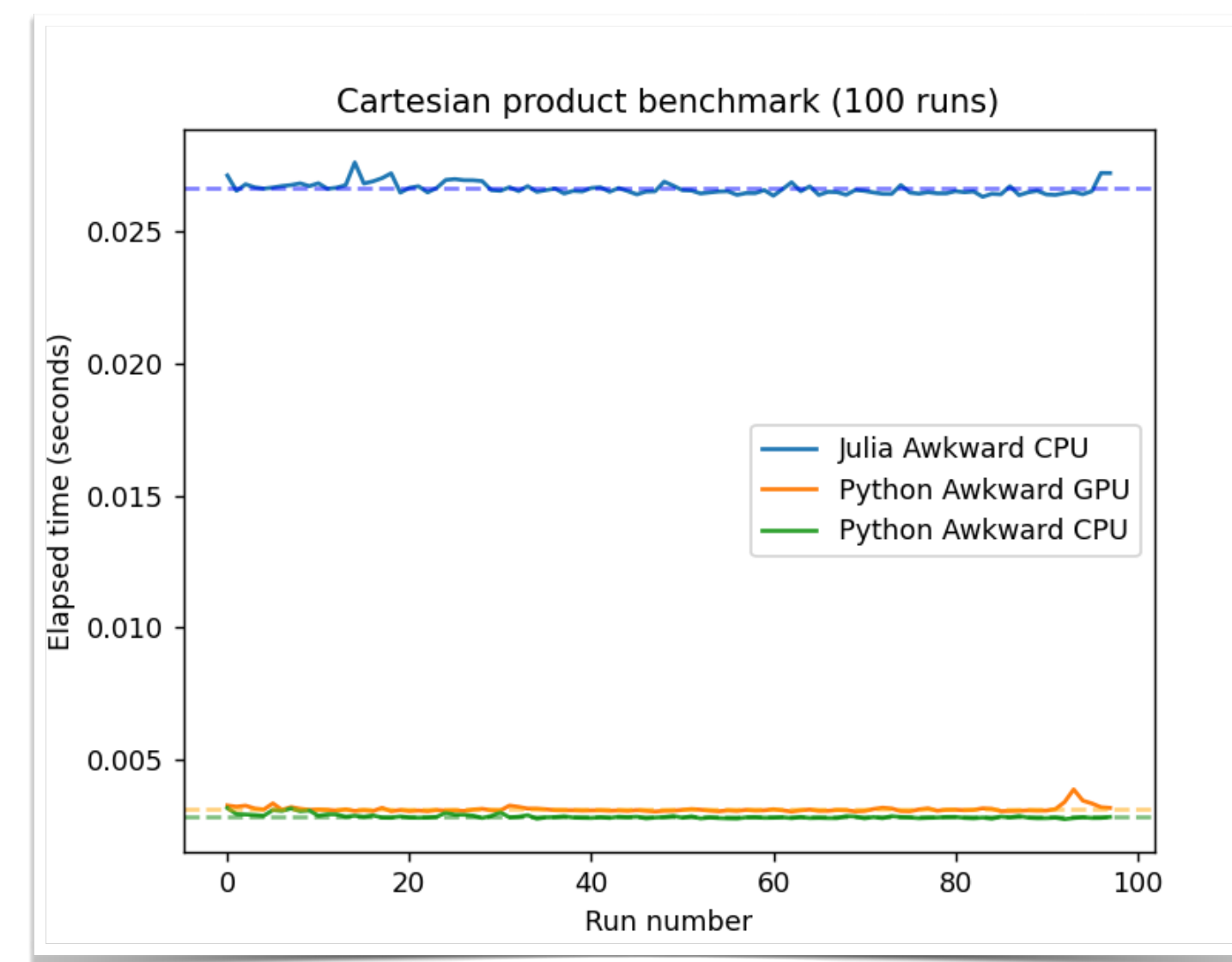
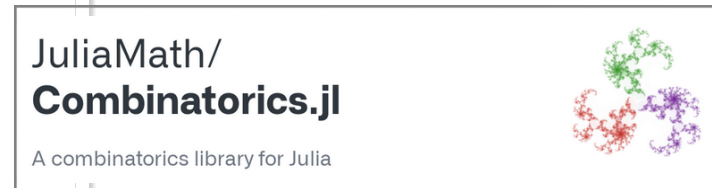
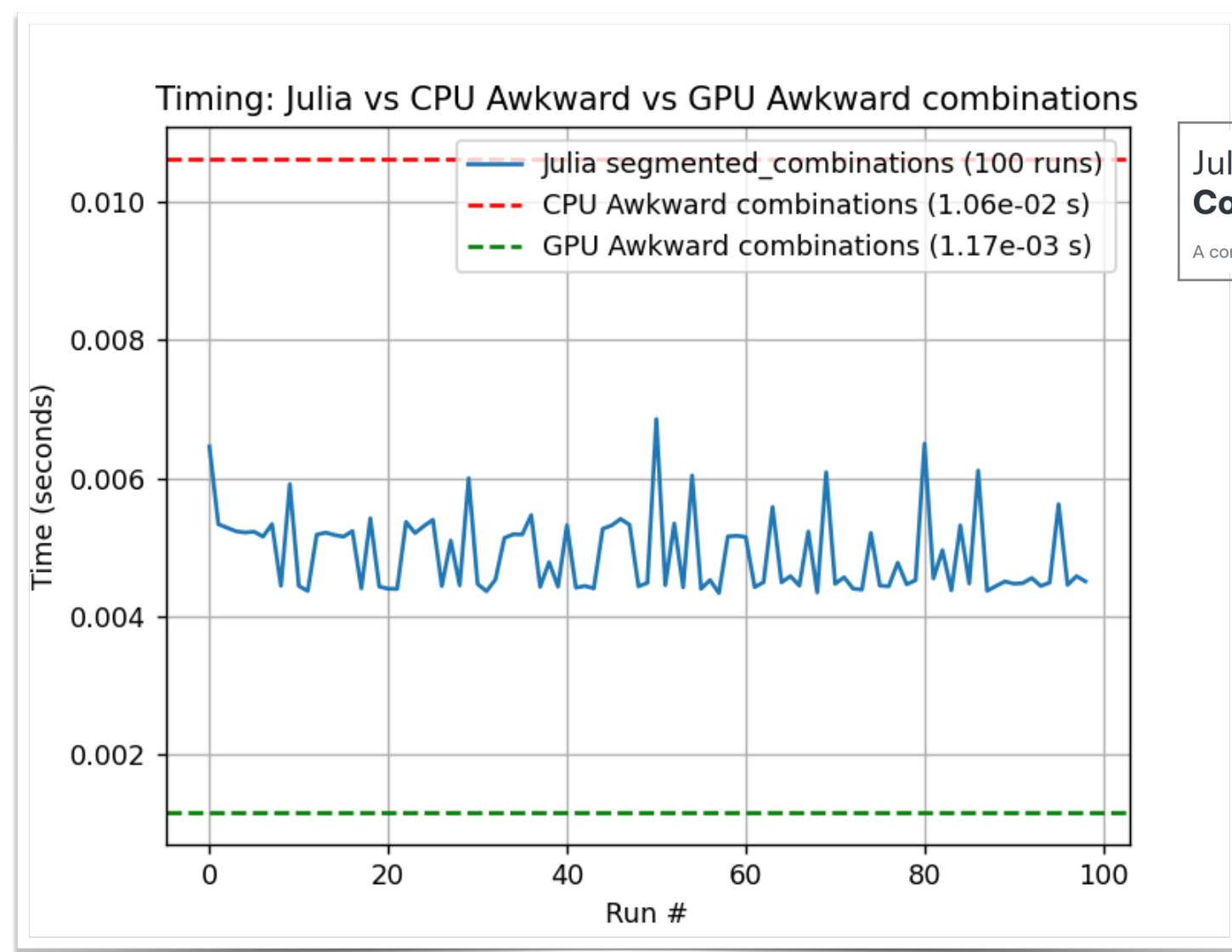
- CHEP 2024: evaluated integration by comparing performance and usability using Julia-native types like Vector of Vectors, which provide similar capabilities for representing nested, irregular data.
 - no significant overhead when combining the two ecosystems.
 - user interest in exploring Julia kernels for Awkward Array: combinatorics.
- Explored Julia's interoperability and its native packages to further optimize HEP computations, offering a practical pathway for gradually introducing Julia to the Python-based HEP community.



Performance Comparison

Encouraging results

- Julia offers a noticeable speedup with less boilerplate code.
- This is the combinations performance plot, comparing the ak.combinations function with a Julia kernel on an awkward array. The Julia kernel uses JuliaMath/Combinatorics.jl.





Summary and discussion

- JuliaHEP, with its emerging set of tools and libraries, aims to facilitate HEP data analysis in Julia. However, given the long timescales of HEP experiments and the substantial legacy software they depend on, introducing Julia alongside existing C++ and Python ecosystems inevitably creates a three-language problem — though this is intended as a transitional, rather than permanent, arrangement.
- To navigate this, three main development strategies have emerged within the JuliaHEP community.
- AwkwardArray.jl backend addresses one of the key challenges in multi-language environments: data sharing across language boundaries. By exposing the underlying data buffers directly to Julia, AwkwardArray.jl avoids data copying and offers a gentle, incremental path for physicists to explore Julia’s capabilities within their existing Python workflows.