IRIS-HEP FELLOW PROJECT: HISTOGRAMS USING NAMBA

Shuo Liu (Fellow Candidate)
sl4921@columbia.edu

Henry Schreiner (Mentor)
henryfs@princeton.edu

Hans Dembinski (Mentor)
hans.dembinski@gmail.com

ABSTRACT

Recent developments in Scikit-HEP libraries have enabled efficient histogramming powered by boost-histogram and fitting into a larger ecosystem of users. Numba is a high-performance Python compiler that uses the industry-standard LLVM compiler library. To enable a fully Numba-enabled event loop for analyses, histogramming step needs to be implemented. In this summer, I will investigate ways to enable boost-histogram's histogramming fill from inside the LLVM Numba loop without stepping through Python.

1 PROPOSAL

In the High Energy Physics community, histogramming is vital to most of our analysis. As part of building tools in Python to provide a friendly and powerful alternative to the ROOT C++ analysis stack, histogramming was targeted as an area in the Python ecosystem that needed significant improvement. Recent developments in Scikit-HEP libraries have enabled fast, efficient histogramming powered by boost-histogram (git) using the hist library and fitting into a larger ecosystem of users.

Numba (git) translates Python functions to optimized machine code at runtime using the industry-standard LLVM compiler library. One key feature in enabling a fully Numba-enabled event loop for analyses is the histogramming step - most loops read (awkward git) data, perform operations (including with vector git), and then fill a histogram. The awkward and vector portions are developed or mostly developed, leaving the histogramming step as the one element missing from a fully Numba enabled event loop. This project will investigate ways to enable a fill from inside the LLVM Numba loop without stepping through Python, to provide first-class Numba support for boost-histogram.

This project has profound implications to facilitate the research of high energy physics computation: by supporting the Numba-enabled hist fill, we were able to make a faster event loop, thereby improving the performance of all projects that rely on boost-histogram, such as hist (git).

2 TIMELINE

The fellow is supposed to start in June and is expected to end in August. My proposed full-time equivalent effort fractions by month are shown as follows.

- June: In this month, I will figure out how Numba works and how it is used in awkward and vector. After that, I will think about how to make the whole workflow for event loop to work together and make a minimal working example notebook that demonstrates a fillable, very simple histogram.
- July: I will go to implement the Numba-enabled fill: to target at simple fills first, then to add weighted and sampled fills later in the month. Besides, some test cases are needed to verify whether this feature works with other components in the event loop.
- August: If things go smoothly, I will create API and add some examples to the documentation this month. I will also write a report or make a slide to summarize my work and list some future works.

APPENDIX

Fig. 1 shows an example that we want to achieve in this project (discussion #552).

```
@nb.njit
def compute_masses(array):
    out = np.empty(len(array), np.float64)
    for i, event in enumerate(array):
        total = vector.obj(px=0.0, py=0.0, pz=0.0, E=0.0)
        for vec in event:
            total = total + vec
        out[i] = total.mass
    return out

out = compute_masses(awkarray)
hist.fill(out)
```

Figure 1: An example we want to achieve.