IRIS-HEP Fellowship Proposal

Pratyush Das

Institute of Engineering & Management, India

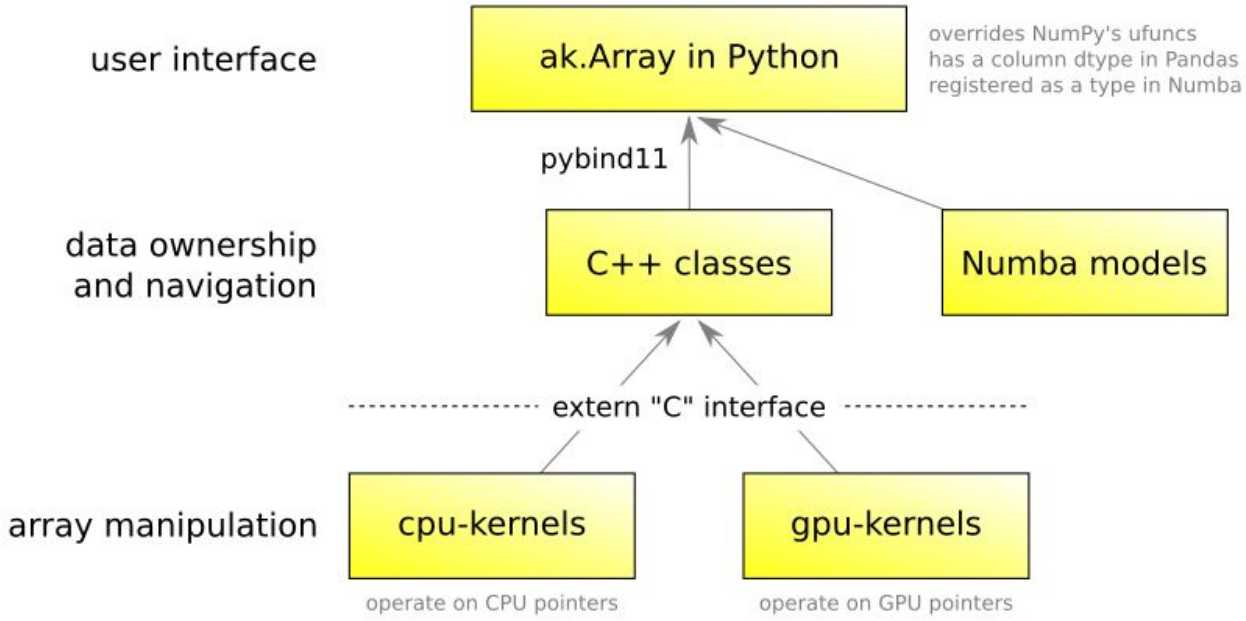
Duration: 1st June, 2020 to 24th August, 2020

Mentor: Jim Pivarski

Translating scalar CPU kernels to vectorized GPU kernels for the Awkward Array library

The Awkward Array library provides a NumPy-like interface to the nested and unequal-length datasets that are often required by particle physicists. The open source project that I will be working on is a C++ (with Pybind11 wrappers) reimplementation of the library written solely in Python that was first released in September 2018.

In Awkward Array's design schematic, the IRIS-HEP fellowship project will involve implementing the gpu-kernels block.



The work would involve creating a library of Awkward-Array GPU kernels preceded by an investigation into the most appropriate way to translate pre-existing CPU kernels to GPU kernels, with an emphasis on generalizing the translation between the scalar code in the currently existing CPU kernels into vectorized code to be executed on GPUs.

A GPU backend would not only allow exponentially faster operations in certain cases but would also allow the arrays to live on the GPU for machine learning. The Awkward Array library is particularly suited for a GPU backend due to its array-at-a-time approach that is common in the design of parallel algorithms which run on GPUs. The main complexity of the project is designing efficient parallel GPU algorithms for the corresponding sequential CPU algorithms. Some of the CPU kernels are embarrassingly parallel and can be naively re-written for the GPU, whereas there are others which have loop carried dependencies and their GPU implementation involves designing algorithms from scratch (or modifying existing algorithms such as the prefix sum algorithm that can be extrapolated to some of the required GPU kernels).

At the end of the summer, users of Awkward Array should be able to naively switch between the CPU and GPU backends without having to write specialized code or even leaving the Python prompt.

Proposed Timeline

Week 1:

- Getting familiar with the Awkward Array codebase - which includes studying the previous iteration of the codebase (awkward-0) as well as understanding how projects such as Coffea and uproot depend on the Awkward Array library.
- Comparing hand-written prefix sum algorithms(Blelloch and Hillis-Steele scans) with each other as well as with the Thrust (and CUB?) implementation to decide the best algorithm to emulate in Awkward Array's GPU kernels.

Week 2:

- Surveying the existing CPU kernels and their characteristics to classify them into categories such as "embarrassingly parallel" and "prefix sum". This would provide an insight into how to translate the kernels and analyze if it is possible to create a tool to naively translate most existing and any new CPU kernels to GPU kernels.

Week 3:

- Creating a tool to automatically generate tests for CPU and GPU kernels.

Week 4:

- Adding metadata to each array node in C++ to indicate whether the node's data are in main memory or a GPU, and if so, which one - with the metadata being read only in Python.
- Adding toCPU(n) and toGPU(n) functions to all array nodes in C++ for transferring data.

Week 5:

- Adding a runtime dispatch mechanism to all array nodes, so that they can select a CPU-kernel or a GPU-kernel for each given task, depending on where the array is located.

Week 6:

- Creating a GPU-kernels directory which initially contains direct copies of the CPU kernels, to get the structure set up and to make sure that all the tests pass.
- Modifying the Awkward Array project structure's layer 3 to layer 4 connector so it can use the GPU kernels.

Week 7:

- Replacing the CPU kernels that are embarrassingly parallel with GPU kernels.
- Replacing the CPU kernels whose base is formed by the prefix sum algorithm(translated for the GSoC evaluation task) with GPU kernels.

Week 8-9:

- Designing algorithms for and translating the remaining CPU kernels to GPU kernels.

Week 10:

- Integrating the CUDA option into package management, so that Awkward Array can be installed with or without CUDA availability, and it switches over gracefully if CUDA libraries are not present.

Week 11:

- Making sure the GPU kernels designed over the summer are future proof and the algorithms can be easily adopted by future GPU platforms.

Week 12:

- Performing scaling tests for CPU kernels vs GPU kernels.
- Providing documentation on the new GPU kernel feature designed over the summer.

Reach goal:

- Writing a shift-reduce C++ level parser for high-level types.