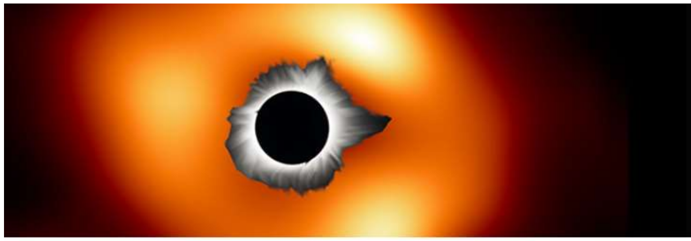

21st MEETING OF THE AAS HIGH ENERGY ASTROPHYSICS DIVISION
HORSESHOE BAY RESORT, HORSESHOE BAY, TEXAS **7-12 APRIL 2024**

Accelerating Compton Imaging of Astrophysical Sources in Python

Jamie Shin[1], Marion Sudvarg[1], Jeremy Buhler[1], James Buckley[1], APT Collaboration[1]
[1]Washington University in St. Louis, Saint Louis, MO

The Python language is an attractive target for computations in astrophysics, as in many other scientific domains. Advantages of Python for these computations include ease of development, strong support for interactive features (e.g., Jupyter notebooks), and an ever-growing library of domain-specific tools built by the scientific community. However, Python's interpreted nature and high level of abstraction can incur substantial performance penalties, even for code that performs mainly numerical operations. We therefore consider the potential of just-in-time compilation (JIT) technologies to permit order-of-magnitude performance improvements for astrophysics computations while remaining entirely in Python.

We investigated potential speedups in code released as part of the first COSI Data Challenge, a Python-based project intended to spur community interest in working with data produced by the Compton Spectrometer and Imager (COSI) mission. The Data Challenge includes Jupyter notebooks that implement Richardson-Lucy deconvolution to image point-source and diffuse Compton-regime gamma-ray emissions, starting with simulated observations from the 2016 COSI balloon experiment. Running these notebooks requires hours of CPU time, which becomes an obstacle for researchers wishing to engage with the Data Challenge. We modified the notebooks to exploit the Numba just-in-time compiler for key computation kernels, as well as the JAXopt optimizer for nonlinear parameter optimization. Although use of these tools requires nontrivial changes to the notebooks' Python code for best performance, they offer the potential for much greater speed and, crucially, parallelization of large matrix operations and optimizer function evaluations across multiple CPU cores. We demonstrate greater than 50-fold speedups for Richardson-Lucy deconvolution on eight CPU cores, as well as substantially faster convergence to a useful image in some cases. We also investigate these tools' potential to deliver speedup using graphics processors (GPUs).