Why Julia?

Ujjwal Panda

: *astrogewgaw.com*

: *github.com/astrogewgaw*

: *ujjwalpanda97@gmail.com*

March 30, 2022

This document uses **Tufte-LaTeX**, LaTeX a class inspired by the works of Edward Tufte. Edward Tufte is a statistician and artist, and Professor Emeritus of Political Science, Statistics, and Computer Science at Yale University. He wrote, designed, and self-published 4 classic books on data visualization.

The Two-Language Problem

The **two-language problem** is well-known in the field of computing. Programmers use **high-level languages** in the initial stages of development, but have to abandon them in the favour of **low-level languages** when deploying their code in production. But *why?* The answer is simple: the former are slow, while the latter are not. The only solution till now has been to write 70% - 90% of the code in a high-level language like **Python**, and then patch up the bottlenecks with a low-level language like **C/C++**. Pulsar astronomy is no exception this trend[1]. In fact, libraries used most often by pulsar astronomers, such as **TEMPO**, **PRESTO**, or **SIGPROC**, are written entirely in **C**. While these codes are reliable and performant, they have proven to be difficult to maintain, extend or document[2].

The problems above have given rise to a disturbing trend: every research group has taken to reinventing the wheel, giving rise to a large number of tools with an overlapping set of features. However, advances in computing offer us a glimmer of hope.

[1] To take an example from our own collaboration, **RIPTIDE** uses a mix of C++ and Python to offer a speedy, high-level interface to the Fast Folding Transform.

[2] The **PINT** project has also raised similar concerns; in fact, the name of the project is a recursive acronym for PINT is not TEMPO3!

Julia: A Possible Solution?

Since its inception in 2012, **Julia** has promised to be the solution to the two-language problem: a high-level language that is as performant as low-level languages. While this may sound too good to be true at first glance, the last ten years have since shown us otherwise. Julia is a dynamically-typed language (like Python). However, it is the only language of that kind that has broken through the petaFLOP barrier: in 2017, the **Celeste** project:

1. Loaded ~178 TB of image data,
2. Produced parameter estimates for 188 million stars and galaxies,
3. Did so in a mere 14.6 minutes,
4. Achieved a speed of 1.5 petaFLOPs,
5. Using 1.3 million threads,
6. On 9300 Knights Landing (KNL) nodes
7. At the Cori supercomputer[3] at NERSC[4].

The project was written in pure Julia, and has since set an example for what can be achieved by pure Julia used at scale. Despite showing such potential, Julia has not gained much attention in the pulsar/FRB astronomy community. I hope to change that via this document, by making a strong case for Julia from a programmer's standpoint.

[3] The sixth fastest supercomputer in the world.

[4] The **N**ational **E**nergy **R**esearch **S**cientific **C**omputing Center (**NERSC**) at Lawrence Berkeley National Laboratory (Berkeley Lab).

Ecosystem

Packaging

Composable

Performance

Interoperability