Jet Reconstruction with Julia

Atell-Yehor Krasnopolski
IRIS-HEP Fellow
Taras Shevchenko
National University of Kyiv

Benedikt Hegner
CERN

Graeme A Stewart
CERN
Jet Reconstruction

- Cluster together measurement points of the same origin
- Algorithm defines the jets
- Iterative process involving computing distances

\[ d_{ij} = \min(k_{ti}^{2p}, k_{tj}^{2p}) \frac{\Delta_{ij}^2}{R^2}, \]

\[ d_{iB} = k_{ti}^{2p}, \]

\[ \Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 \]
Why Julia (Speed)

- Python & C++ → Julia?
- Simple and efficient, way faster than Python
- Fast by design, not because of packages
- JIT-compiled
- Can interact with C, FORTRAN & Python

(towardsdatascience.com/r-vs-python-vs-julia-90456a2bcbab)
Why Julia (Convenience)

```julia
using Plots

# JIT-compiled for `c::ComplexF64`

function mandelbrot(c; maxiter=100)
    z = c
    for n in 1:maxiter
        if abs(z) > 2
            return n-1
        end
        z = z^2 + c
    end
    maxiter
end

C = hcat([x+y*im for x in np.arange(-2, 2, 0.01) for y in np.arange(2, -2, -0.01)])

# broadcasting over C, we get a matrix of integer type
M = ~mandelbrot.(C)

heatmap(M) # plot
```

Jet Reconstruction with Julia
Naive Method

- Naive implementation takes ~42 lines of code
- Not a very sophisticated approach
Naive Method

FastJet is a package written in C++ that is considered a standard.

Atell-Yehor Krasnopolski

Jet Reconstruction with Julia
Julia is extremely easy to profile and optimise because of such native tools.
Changes

- Cache the distances
- Optimise the distance function to avoid floating point powers
- Clean the code up
Better Method

Jet Reconstruction with Julia
Better Method

![Graph showing time vs number of inputs for C++ FastJet and Julia.](image-url)
Next Steps

- Implement an exact copy of FastJet’s optimised version of the algorithm and test if this sophisticated approach is better
- $O(N \log N)$ instead of $O(N^3)$
FastJet’s Tricks

Atell-Yehor Krasnopolski

Jet Reconstruction with Julia
FastJet’s Tricks

It took a lot of effort to dive into the FastJet code and make it work properly.
FastJet’s Tricks

This is another step that was hard to make (a lot of profiling and low level thinking)
The longest bars are simply “for” loops that we cannot optimise.
Additional Features

- Possibility for other sequential recombination algorithms in Julia
- Plotting the results
- Comparison with FastJet (the C++ golden standard)
- Support for user-defined data types and extensions & GPU (because of multiple dispatch)
What Have I Learned?

- A lot about high energy physics (tasks, software, algorithms)
- How to work with HEP data
- Automate tests for Julia repositories
- Use Julia’s profiling tools
Summary

- Julia can be fast if you know what to do
- Julia can be a good alternative to Python
- Still a lot to explore in the usage of Julia in HEP yet
- Conclusion is yet to be made

JetReconstruction.jl
github.com/gojakuch/JetReconstruction.jl

github.com/JuliaHEP