Summary

Just-in-time Compiled Python for Bioinformatics Research

Johanna Elena Schmitz, Jens Zentgraf and Sven Rahmann

ISMB 2024, Montréal, Canada (July 12, 2024)

Learning Objectives

Today we discussed

- the difference between interpretation, lazy and eager/early compilation,
- how numba just-in-time compiles parts of your code,
- when numba can accelerate your code (and when it cannot),
- the differences between compilable and non-compilable Python code,
- how to speed up an initial Python implementation to handle larger data faster,
- how to parallelize Python in spite of the Global Interpreter Lock (GIL) with compiled functions,
- how to build a browser app with streamlit,
- understand how to use NFAs for motif search.

Take-Home Messages

- Python alone is not competitive on large data.
- Numba allows to speed up numeric and textual computations.
- If the code is already structured, it is often enough to just add `@njit`.
- Try to always use NumPy arrays instead of lists or generators.
- Using parameters as compile-time constants is an optimization that is often worth the additional coding effort.
- Parallelization with `parallel=True` and `prange` sometimes helps additionally, but can also make it slower. Only use it in rare circumstances.
- The nopython mode allows to release the GIL, allowing the use of threads and threadpools.
- Streamlit is an easy way to create shareable web apps in python.
- Streamlit support many ways to interact with the user, display text, media or charts and connect to data sources.

Further Reading

Documentation

- Numba https://numba.readthedocs.io/en/stable/
- Streamlit https://streamlit.io/

Tools using `numba`

- Xengsort https://gitlab.com/genomeinformatics/xengsort
- Hackgap https://gitlab.com/rahmannlab/hackgap

Feedback Survey

https://docs.google.com/forms/d/e/1FAIpQLSfrjbxbGdE45OFPQgM58JDt59B–gonwjuKw1HkHrR4FLvhhw/viewform