Study of new dataframe backends for the olefin library

Andrés Navarro Pedregal

What is a dataframe backend?

A library that manages the data and does operations on it

Why matters?

- Defines the speed of the data processing
- The types of data you can store
- How easy it is to fidget around with the data

Requirements

- Performance in the order of seconds
- Lists of variable size inside columns: for new features
- Cross-platform compatible: Windows and Linux
- Export to compressed file formats
- Parallelization of runs

Backends studied

- **Pandas with numpy**: current setup
- **Pandas with pyarrow**: new version of pandas based on arrow
- **Polars**: really fast implementation of arrow focused on parallelization
- **PyROOT**: CERN’s ROOT for python
- **Datatable**: another backend worth to check

Pandas with numpy

Advantages:

- **Current setup, little change**
- Support for older version of python
- Export to any type

Disadvantages:

- **Does not support variable size lists inside dataframes**
- Slow compared to arrow implementation (still fast enough for current use case)

Pandas with pyarrow

Advantages:

- Arrow implementation, fast
- Has schemas, can define the structure in advance -> more robust
- **Can be implemented gradually with the previous versions**
- Export to any type
- **Can operate in lists inside a dataframe**

Disadvantages:

- A bit less performant that polars
- Does not have parallelization built in
- Supported by python 3.8 and up

Polars

Advantages:

- Most performant
- **Parallelization built in with lazy processing**
- Export to any type
- Has schemas same as pandas with pyarrow
- **Can operate in lists inside dataframes**
- Supports zero-copy data sharing

Disadvantages:

- Might be too overkill
- **Need to redo most of the library** but can be done incrementally
- Supported by python 3.8 and up

PyROOT

Advantages:

- **Widespread in the CERN ecosystem**
- Export in compressed formats: TTree, RNTuple
- Good fit functions for the data

Disadvantages:

- **Need to rewrite the whole library**
- **Not support for some functions used:** lists inside columns
- No parallelization
- **No backwards compatibility with previous versions**

Datatable

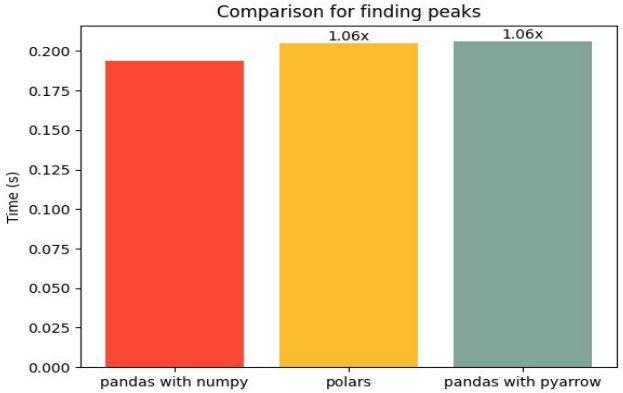
Advantages:

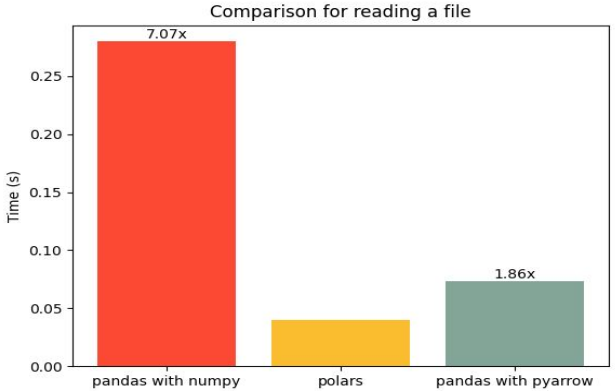
- Empty :)

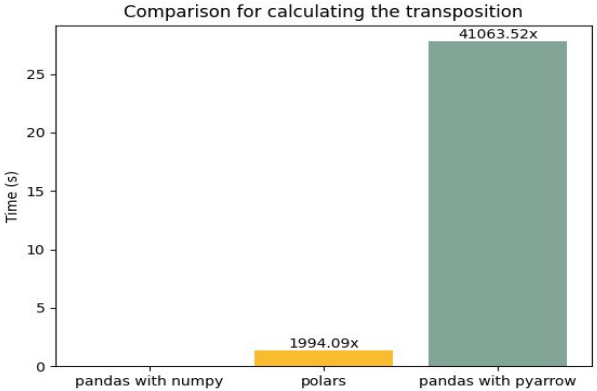
Disadvantages:

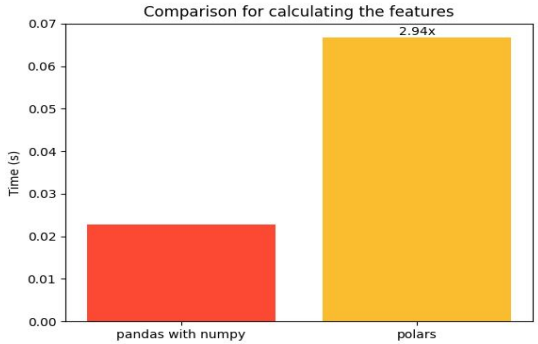
- Less performant than arrow
- **Need to rewrite the whole library**
- **Does not support lists inside columns**
- No parallelization

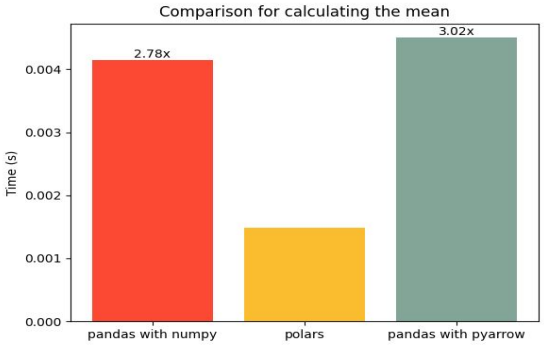
Tests











- Pandas with pyarrow and polars are usually faster for dataframe operations
- But Arrow format is really bad at transposing data!

Conclusions

All in all, if we want a more performant library sustainable for the future; I would recommend two options: **pandas with pyarrow (as it is the new standard for pandas) or polars.**

Moreover, they can be interchanged with the previous version so **backwards compatibility can be kept**.

Pandas with pyarrow:

- Easiest, **will take less time**

Polars:

- If we want the **best performance** and utilization of the computers.

Problem: need to drop python 3.7