Pandarallel — A simple and efficient tool to parallelize your pandas computation on all your CPUs

How to significantly speed up your pandas computation with only one line of code.

Manu NALEPA

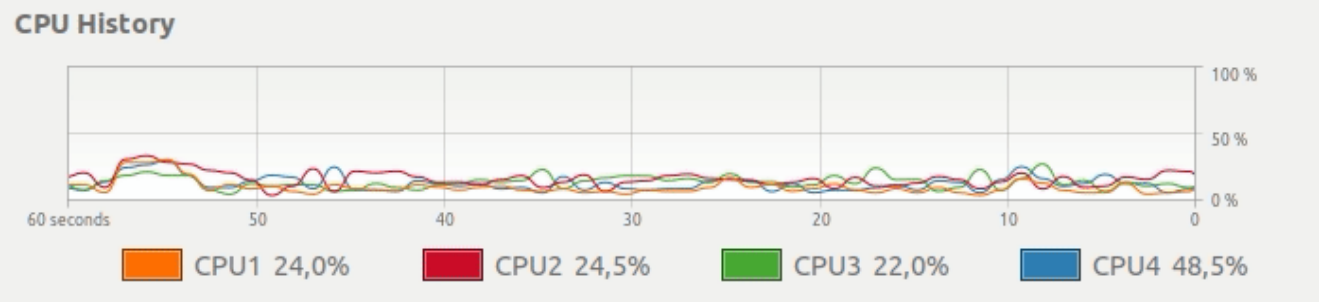
Complete **Pandaral·lel** repository and documentation is available on this GitHub page.

The library presented in this post is only supported on Linux & MacOS.

What issue does bother us?

With pandas, when you run the following line:

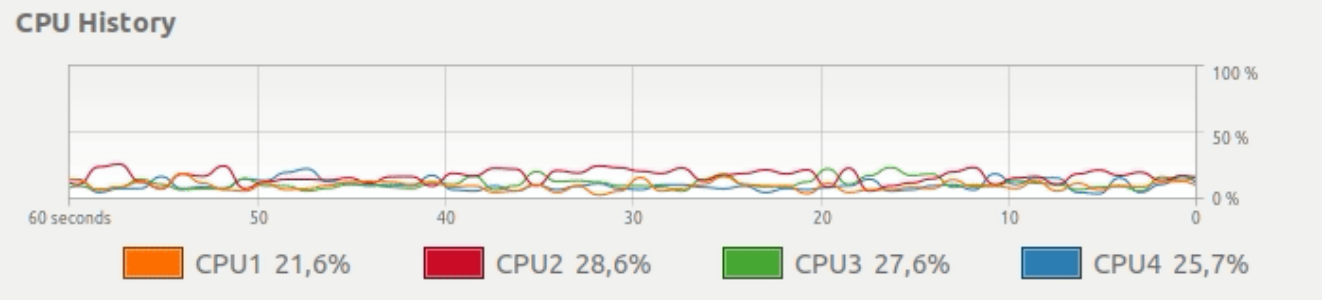
You get this CPU usage:



Standard Pandas apply — Only 1 CPU is used.

Even if your computer has several CPUs, only one is fully dedicated to your calculation.

Instead of this CPU usage, we would like **a simple way** to get something like this:



Parallel Pandas apply — All CPUs are used.

How Pandaral·lel helps to solve this issue?

The idea of **Pandaral·lel** is to distribute your pandas calculation over all available CPUs on your computer to get a significant speed increase.

Installation:

On Windows, **Pandaral·lel** will works only if the Python session (`python`, `ipython`, `jupyter notebook`, `jupyter lab`, ...) is executed from Windows Subsystem for Linux (WSL).

On Linux & macOS, nothing special has to be done.

Import & Initialization:

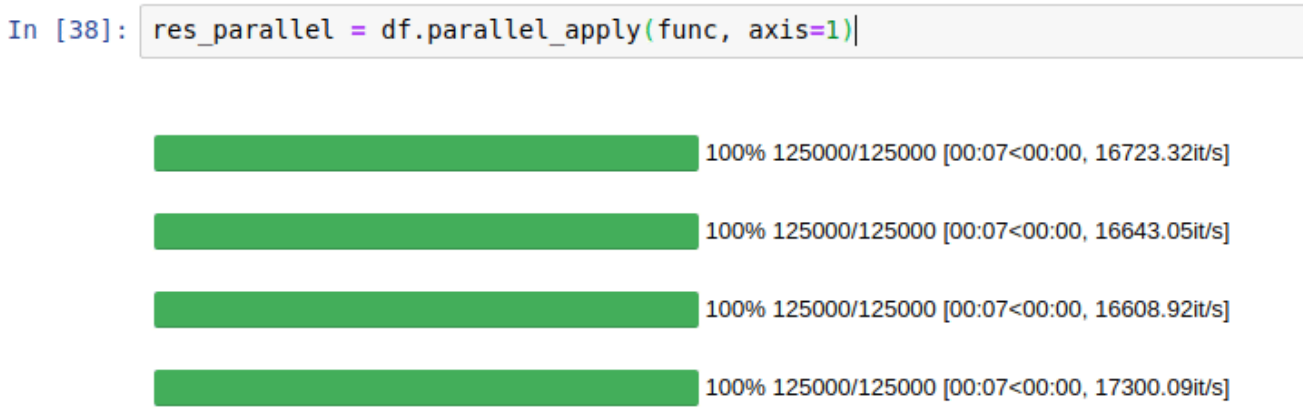
Usage:

With a simple use case with a pandas DataFrame `df` and a function to apply `func`, just replace the classic `apply` by `parallel_apply` .

And you'r done!

Note that you can still use the classic `apply` method if you don't want to parallelize computation.

You can also display one progress bar per working CPU by passing `progress_bar=True` in the `initialize` function.



Parallel apply with a progress bar

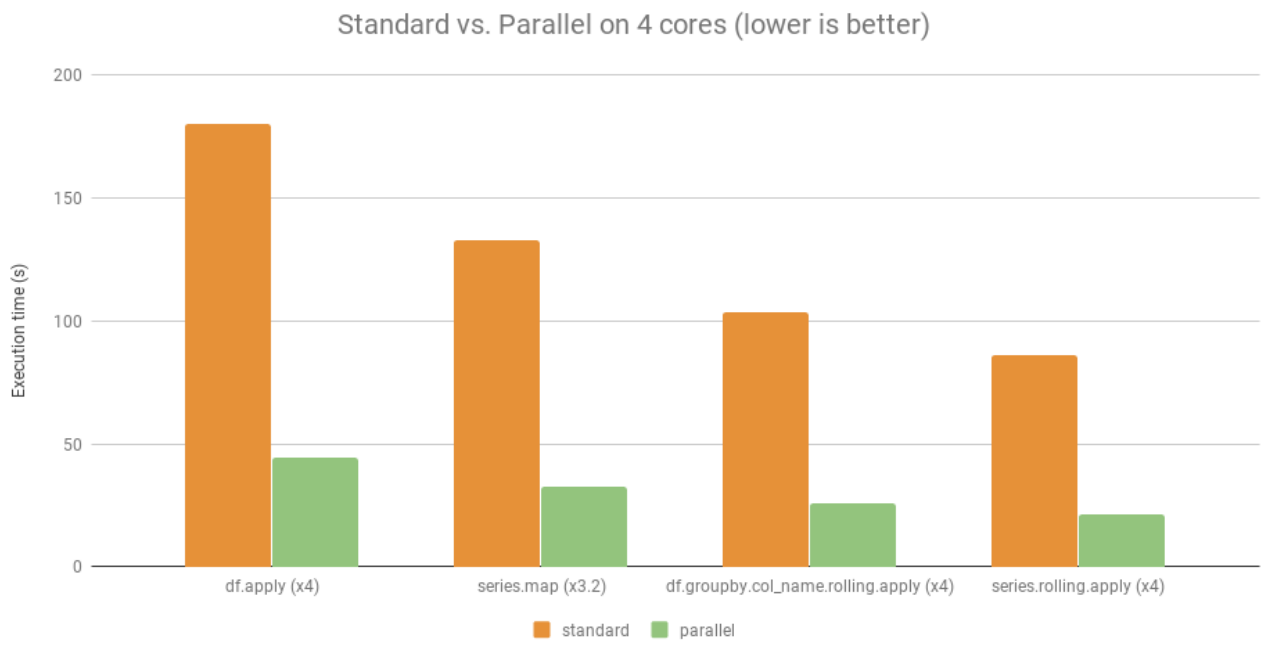
And with a more complicated use case with a pandas DataFrame `df`, two columns of this DataFrame `column1` and `column2`, and a function to apply`func`:

Benchmark

For four of the examples available here, on the following configuration:

- OS: Linux Ubuntu 16.04

- Hardware: Intel Core i7 @ 3.40 GHz — 4 cores



Standard vs. Parallel on 4 cores (lower is better)

Except for `df.groupby.col_name.rolling.apply`, where speed increases only by a x3.2 factor, the average speed increases by about x4 factor, which is the number of cores on the used computer.

How does it work under the hood?

When `parallel_apply` is called, **Pandaral·lel:**

- instantiates a Pyarrow Plasma shared memory, then
- creates one sub processes for each CPU, and asks each CPU to work on a sub part of the DataFrame, then
- combine all the results in the parent process

The main advantage of using a shared memory compared

to other inter-process communication medium is that there is no serialization/de-serialization which can be very CPU expansive.

If you find this tool useful but if a feature is missing, please write a new feature request here.