DATA
WORKSHOP

DW Poznań - Duże zbiory danych #9

2019-05-21
Czyli jak ogarnąć dane New York City Crimes

Agenda

01. **New York City Crimes -** dane

02. **Csv, pickle, hdf5, feathrow, pyarrow - jak zapisywać dane i odczytywać.**

03. **Dask, MODIN, CUdf**

04. Obliczenia - i porównania

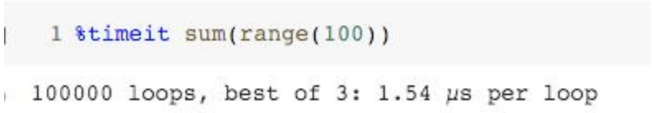
New York City crimes

https://data.cityofnewyork.us/Public-Safety/NYPD-Complaint-Data-Historic/qgea-i56i

- Dane zawierają informacje na temat przestępczości w Nowym Jorku (New York City Police Department (NYPD)) za okres 2006 do końca 2017
- Mają 7,309,655 rekordów
- Plik .csv ma ponad 2 GB

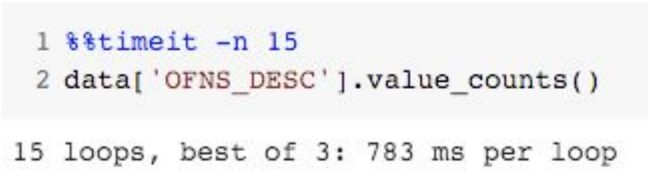
Odczyt performance %timeit, %time

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/01.07-Timing-and-Profiling.ipynb



```
%timeit sum(range(100))
```

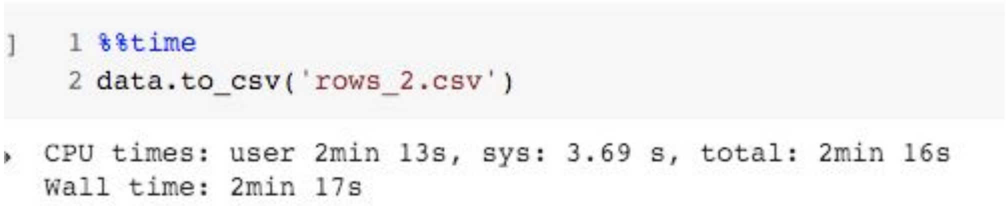
```
100000 loops, best of 3: 1.54 µs per loop
```



```
%%timeit -n 15
data['OFNS_DESC'].value_counts()
```

```
15 loops, best of 3: 783 ms per loop
```

Uruchomienie kilka razy i wyliczenie średniej czasu uruchomienia



```
%%time
data.to_csv('rows_2.csv')
```

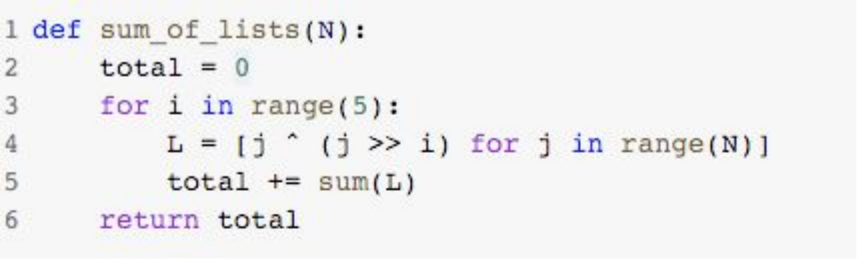
```
CPU times: user 2min 13s, sys: 3.69 s, total: 2min 16s
Wall time: 2min 17s
```

Wyliczenie czasu uruchomienia komórki

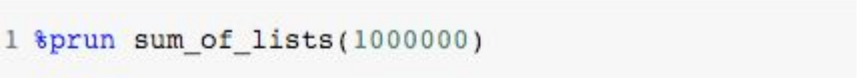
%PRUN

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/01.07-Timing-and-Profiling.ipynb

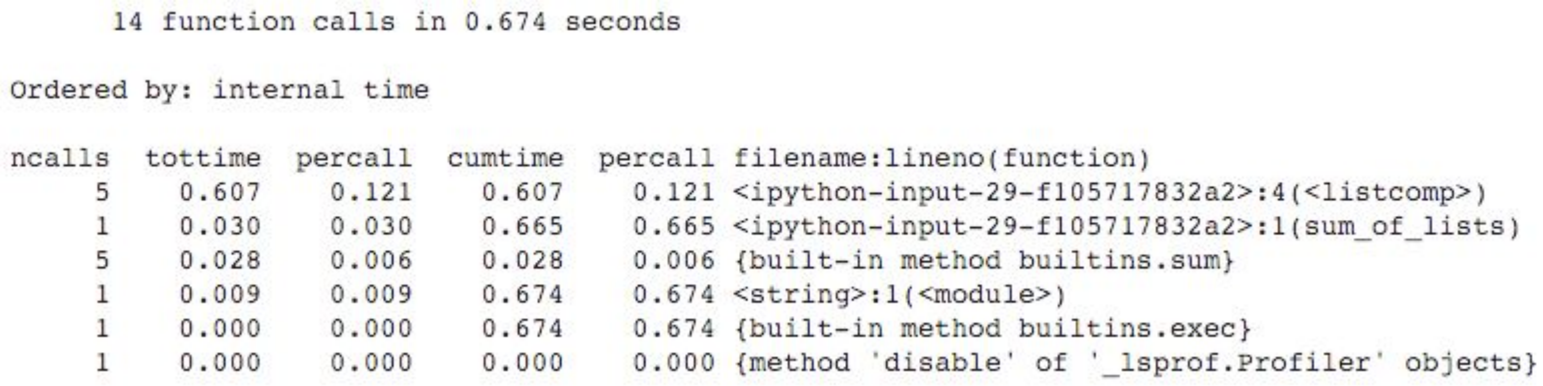
Wyliczenie czasu uruchomienia pojedynczych wierszy



```python
def sum_of_lists(N):
    total = 0
    for i in range(5):
        L = [j ^ (j >> i) for j in range(N)]
        total += sum(L)
    return total
```



```python
%prun sum_of_lists(1000000)
```

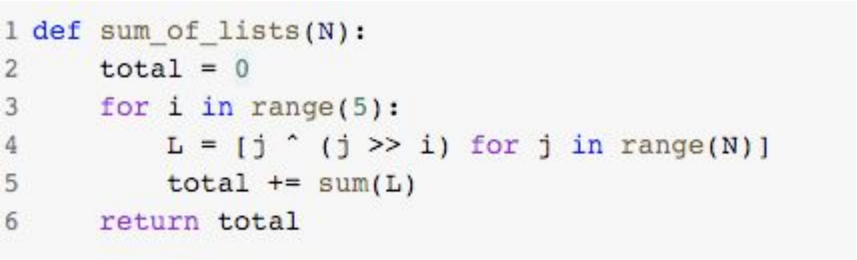


```
         14 function calls in 0.674 seconds

   Ordered by: internal time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        5    0.607    0.121    0.607    0.121 <ipython-input-29-f105717832a2>:4(<listcomp>)
        1    0.030    0.030    0.665    0.665 <ipython-input-29-f105717832a2>:1(sum_of_lists)
        5    0.028    0.006    0.028    0.006 {built-in method builtins.sum}
        1    0.009    0.009    0.674    0.674 <string>:1(<module>)
        1    0.000    0.000    0.674    0.674 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

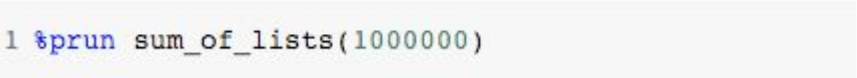
Memory_profiler %memit

https://pypi.org/project/memory-profiler/

Wyliczenie pamięć użytą podczas operacji



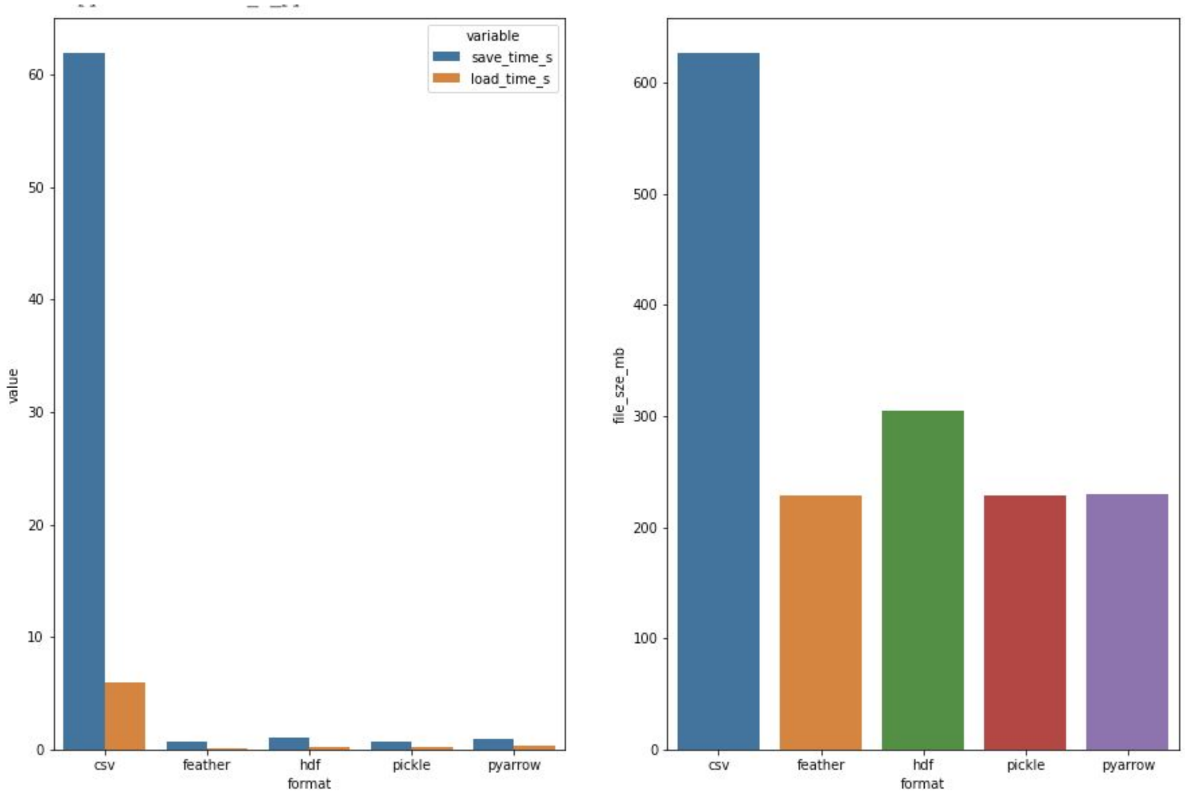
```python
def sum_of_lists(N):
    total = 0
    for i in range(5):
        L = [j ^ (j >> i) for j in range(N)]
        total += sum(L)
    return total
```



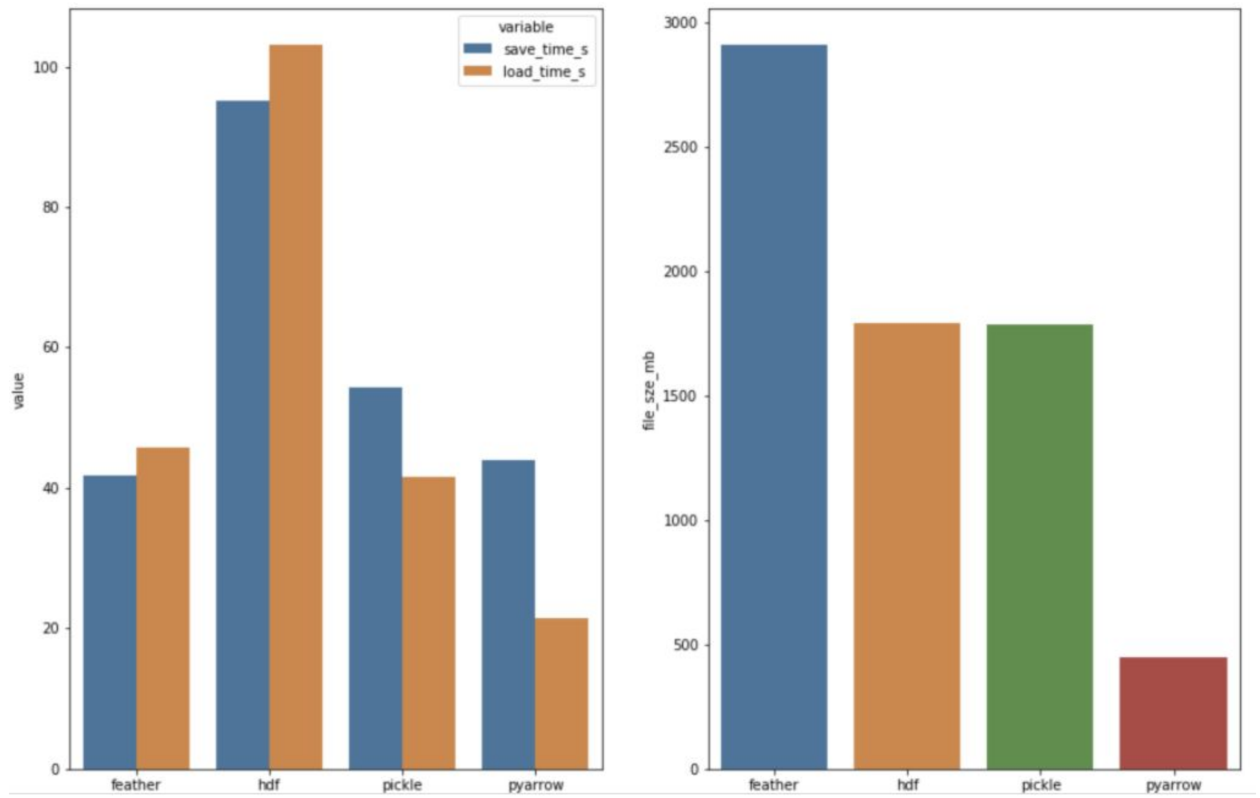
```python
%prun sum_of_lists(1000000)
```

Csv - DataFrame: 10000000 rows

https://towardsdatascience.com/the-best-format-to-save-pandas-data-414dca023e0d



Pickle, hdf5, feathrow, PyArrow - NYC Data



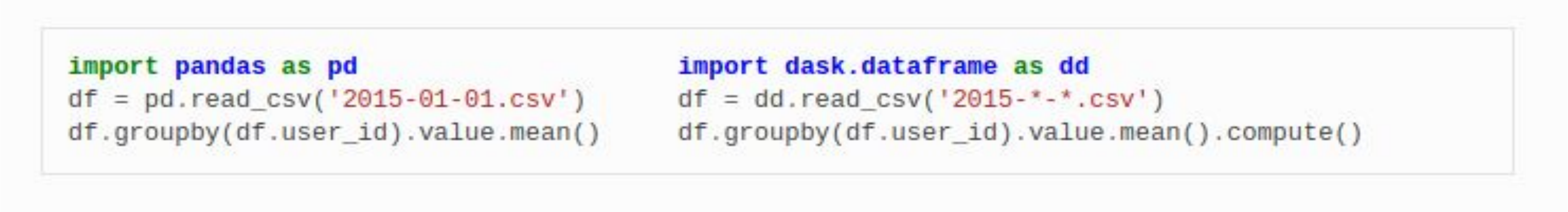
Dask

Dask is a flexible library for parallel computing in Python.

Dask is composed of two parts:

1. **Dynamic task scheduling** optimized for computation. This is similar to *Airflow, Luigi, Celery, or Make*, but optimized for interactive computational workloads.
2. **“Big Data” collections** like parallel arrays, dataframes, and lists that extend common interfaces like *NumPy, Pandas, or Python iterators* to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers.

https://docs.dask.org/en/latest/

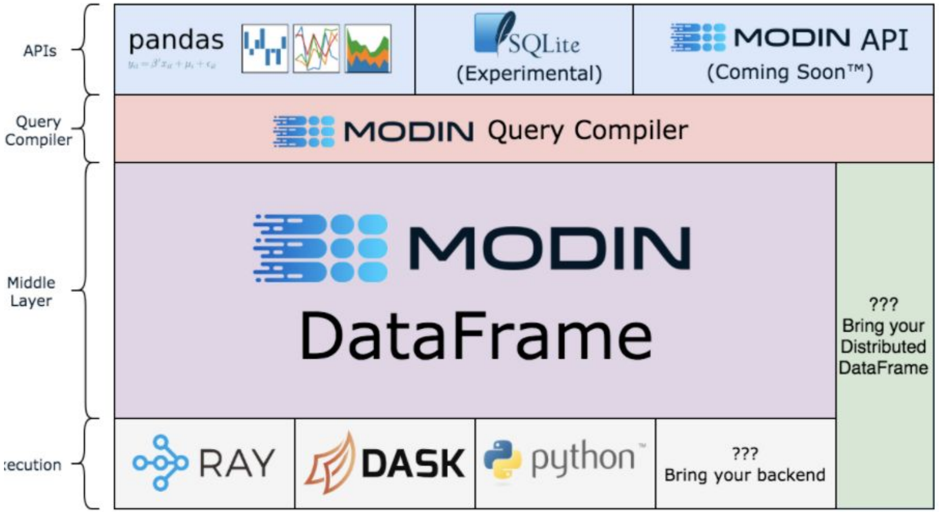


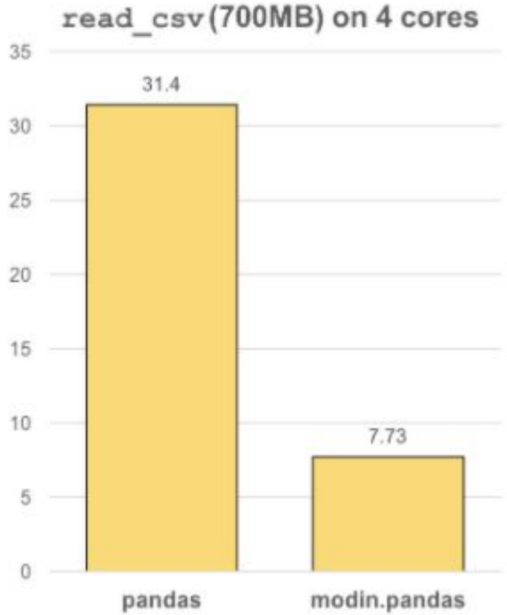
```
import pandas as pd                      import dask.dataframe as dd
df = pd.read_csv('2015-01-01.csv')       df = dd.read_csv('2015-*-*.csv')
df.groupby(df.user_id).value.mean()      df.groupby(df.user_id).value.mean().compute()
```

MODIN

https://github.com/modin-project/modin
https://morioh.com/p/2fd9a93b6e53 - compare CUDF and MODIN
https://towardsdatascience.com/get-faster-pandas-with-modin-even-on-your-laptops-b527a2eeda74 - COMPARE MODIN AND PANDAS

- Jest to multiprocesorowy Dataframe który może przyspieszyć wiele operacji.



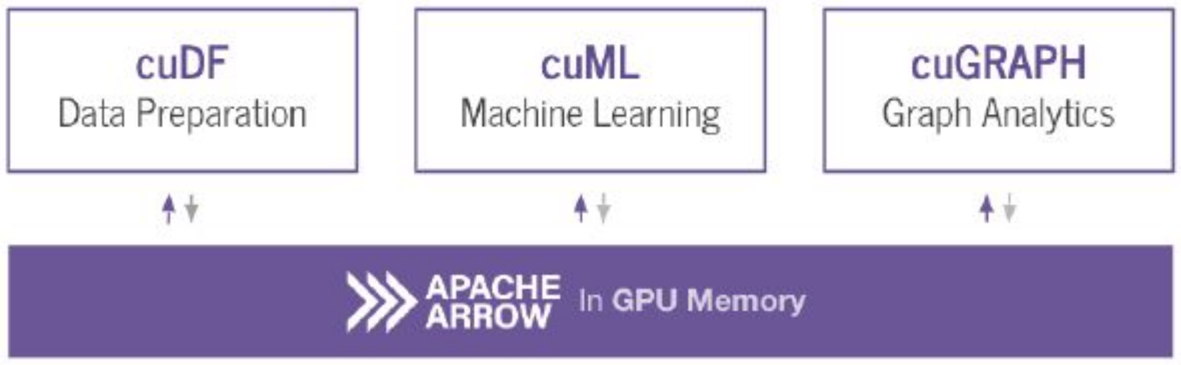


CUDF

https://rapids.ai/start.html
https://morioh.com/p/2fd9a93b6e53 - compare CUDF and MODIN
https://devblogs.nvidia.com/gpu-accelerated-analytics-rapids/

https://towardsdatascience.com/faster-pandas-with-parallel-processing-cudf-vs-modin-f2318c594084

- Biblioteka która cały DataFrame wysyła do pamięci GPU gdzie jest przetwarzany.
- Wszystko przez parsowanie CSV do parsowania jest robione po stronie GPU



Kolejne kroki

- Spotkanie za 2 tygodnie
- Projekt

https://github.com/dataworkshop/dw-poznan-project

Dziękuję