Project Name: Reading Large Files Within Few Seconds Using VAEX Library

Table of Content

Demo
Overview
Motivation
Technical Aspect
Installation
Run/How to Use/Steps
Directory Tree/Structure of Project
To Do/Future Scope
Technologies Used/System Requirement/Tech Stack
Credits

Demo

In [67]:
```
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Columns: 500 entries, col0 to col499
dtypes: int32(500)
memory usage: 1.9 GB
```

Creating Csv files

In [68]:
```
# Creating my own csv file to store dataframe's data that I have created.
file_path = 'final_data.csv'
df.to_csv(file_path, index=False)
```

Create Hdf5 files

In [ ]:
```
# Hierarchical Data Format is a set of file formats designed to store and organize large amounts of data.
# from_csv means read a csv file as dataframe and optionally convert to an hdf5 file.
```

In [81]:
```
vaex_df = vaex.from_csv(file_path, convert=True, chunk_size=5_000_000)
```

In [82]:
```
type(vaex_df)
```

Out[82]:
```
vaex.hdf5.dataset.Hdf5MemoryMapped
```

In [101]:
```
vaex_df.head()
```

Out[101]:

#	col0	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19	col20	col21	col22	col23
0	94	63	26	8	83	10	89	75	39	57	40	34	65	89	88	97	6	48	24	39	85	0	78	57
1	72	33	73	61	50	65	78	31	93	56	89	97	95	82	13	34	74	0	67	26	9	30	77	5
2	9	48	50	23	96	93	97	70	33	43	25	91	63	93	26	27	8	31	13	19	7	13	79	75
3	69	47	95	65	75	0	35	35	42	65	99	63	40	84	20	86	69	99	54	56	53	88	18	63
4	18	94	45	95	3	0	71	85	89	42	22	6	35	87	3	3	54	76	98	76	40	88	41	47
5	7	95	73	5	10	26	1	32	97	63	6	46	75	23	81	10	24	44	69	34	44	93	25	0
6	93	98	36	25	84	34	56	86	28	10	41	20	61	28	62	62	41	97	65	52	92	0	65	11
7	67	42	10	0	28	27	9	25	48	80	79	26	2	57	69	86	89	26	6	94	42	15	85	78
8	13	83	67	17	58	74	7	90	25	79	19	94	68	19	83	65	38	33	22	21	8	59	9	30
9	55	88	6	6	89	59	68	43	52	52	12	83	63	95	70	56	19	57	86	83	8	36	66	98

Expression system

Don't waste memory or time with feature engineering, we (lazily) transform your data when needed.

In [107]:
```
%%time
vaex_df['multiplication_col13']=vaex_df.col1*vaex_df.col3
```

```
Wall time: 0 ns
```

Out-of-core DataFrame

Filtering and evaluating expressions will not waste memory by making copies; the data is kept untouched on disk, and will be streamed only when needed. Delay the time before you need a cluster.

In [109]:
```
vaex_df[vaex_df.col2>70]
```

Out[109]:

#	col0	col1	col2	col3	col4	col5	col6	col7	col8	col9	col10	col11	col12	col13	col14	col15	col16	col17	col18	col19	col20	col21	col22
0	72	33	73	61	50	65	78	31	93	56	89	97	95	82	13	34	74	0	67	26	9	30	77
1	69	47	95	65	75	0	35	35	42	65	99	63	40	84	20	86	69	99	54	56	53	88	18
2	7	95	73	5	10	26	1	32	97	63	6	46	75	23	81	10	24	44	69	34	44	93	25
3	66	56	77	56	72	88	46	26	58	43	26	16	90	68	5	59	2	64	62	23	5	18	13
4	89	0	97	89	34	8	33	20	32	18	39	93	25	24	9	81	42	35	87	89	89	33	50
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
289,468	91	66	87	32	89	93	96	37	41	8	48	69	91	85	69	46	5	45	34	19	68	47	56
289,469	65	42	87	49	15	67	75	63	70	89	32	46	99	65	97	94	10	14	56	59	11	89	34
289,470	71	0	87	63	0	75	13	37	88	92	40	82	74	19	70	45	52	59	49	85	98	40	74
289,471	93	0	92	1	16	64	25	15	40	38	87	41	6	54	65	22	90	52	0	64	51	68	11
289,472	62	93	73	81	76	21	23	2	69	80	48	39	54	1	63	5	15	88	35	91	47	2	33

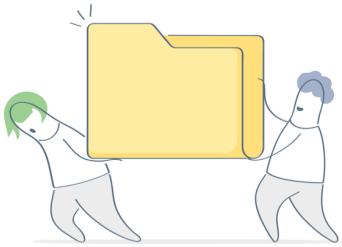
In [119]:
```
%%time
vaex_df.groupby(vaex_df.col1,agg='count')
```

```
Wall time: 25.1 ms
```

Out[119]:

#	col1	count
0	63	10088
1	33	9864
2	48	10207
3	47	9935
4	94	9977
...	...	...
95	73	10107
96	76	10202
97	28	10072
98	77	9861
99	69	10177

Overview



This is about reading large files with vaex library which wastes less memory.
There is a stark difference between large and big data.
It used Pandas, Numpy and vaex libraries.
These libraries help to perform individually one particular functionality.
Data is unavoidably messy in real world.
And Pandas, is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data. Pandas objects rely heavily on Numpy objects.
Numpy is used for working with arrays. It stands for Numerical Python.
Vaex is used to calculate statistics.
Data Science professionals often encounter very large data sets with hundreds of dimensions and millions of observations. So, it is one of the important skills that I am learning here. There are multiple ways to handle large data sets. Here, I am using another library than pandas for the task. It supplies precisely what we need.
Parameter essentially means the number of rows to be read into a data frame at any single time in order to fit into the local memory.
Here, we are loading only some of the lines into memory at any given time. By doing this, basically we have reduced memory usage and still receive same results.
The screenshot will help you to understand flow of output.

Motivation

One of the major motives to create this, is I wanted to gain knowledge with other than common libraries of Python. The reason behind making is, I was baffled when I encountered an error and I couldn't read the data from csv file as my local machine has 8GB of RAM. Therefore, thought to create this one. The purpose of creating this repository is I wanted to dig deeper into other python's library, so after few hours of research I got to know about vaex library which can perform operations on large files. Though here, I have not even used that big dataset file. As I was more interested in concept rather than anything else. By building such mini project helped me to gain knowledge about other python's library which is relatively new to use. Reading files whether small or big or large is basic starting point of any project which every Data Professional should know about. So at least by reading data, they can start to analyse it and not just be dependent. This can speed things up and that is why we want. This expertise with experience can bridge gap between digital marketing and data science.

Technical Aspect

Vaex is a high-performance Python library for lazy Out-of-Core Data Frames (similar to Pandas), to visualize and explore big tabular datasets. It calculates statistics such as mean, sum, count, standard deviation etc, on an N-dimensional grid for more than a billion (10^9) samples/rows per second. Visualization is done using histograms, density plots and 3d volume rendering, allowing interactive exploration of big data. Vaex uses memory mapping, zero memory copy policy and lazy computations for best performance (no memory wasted).
Numpy contains a multi-dimensional array and matrix data structures. It works with the numerical data. Numpy is faster because is densely packed in memory due to its homogeneous type. It also frees the memory faster.
Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.
Pandas is very efficient with small data (usually from 100MB upto 1GB) and performance is rarely a concern. Pandas has its own limitation when it comes to big data due to its algorithm and local memory constraints.
hdf5 means Hierarchical Data Format is a set of file formats designed to store and organize large amounts of data.

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed then please install Anaconda Navigator from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.
Note: Preferred to Perform this project in Google Colab.

Open Anaconda Prompt, Perform the following steps:

cd <PATH>
pip install vaex
pip install numpy
pip install pandas

You can also create requirement.txt file as, pip freeze > requirements.txt

run files.

If you want you can directly install packages into Jupyter Notebook Cell with ! in front.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.
eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\5)Reading_Large_Data\Reading_Large_Files_VAEX
In Anconda Prompt, pip install -r requirements.txt to install all packages.
Open in Jupyter Notebook, <filename>.ipynb
That is,
Open in Jupyter Notebook,
Reading_Large_Dataset_Within_Few_Seconds_Using_VAEX_Library.ipynb
Then it creates final_data.csv is created of 2GB, it is output file.
And also, final_data.csv.hdf5 is created of 4GB, it is output file.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder] [cd means change directory]

Directory Tree/Structure of Project

Folder: 5)Reading_Large_Data>Reading_Large_Files_VAEX
Reading_Large_Dataset_Within_Few_Seconds_Using_VAEX_Library.ipynb

To Do/Future Scope

Can try with Image Dataset same library.

Technologies Used/System Requirement/Tech Stack

Credits

Krish Naik Channel