Published in SFU Professional Computer Science

Data Miners
Feb 10 · 11 min read · Listen

Save

Adios Pandas! Process Big Data in a Flash using Terality, Dask, or PySpark

Check out this blog to operate large datasets with ease by changing just the import line!

Authors: *Dhruv Patel, Himalya Bachwani, Kishan Thumar, Rahil Balar*

This blog is written and maintained by students in the Master of Science in Professional Computer Science Program at Simon Fraser University as part of their course credit. To learn more about this unique program, please visit {sfu.ca/computing/mpcs}.

You should use Pandas, it is super fast and convenient
Just change the import line and make it 100x faster
imgflip.com

Motivation

Don’t you always love Pandas? As a data scientist, you are always told to use/learn Pandas. It is the go-to library because of its easy data representation, hassle-proof syntax, and flexibility.

> *“Slow and steady wins the race, but not in Big Data” — Data Miners.*

Here in the case of Big Data, fast and reliable wins the competition. No doubt pandas have many extensive features but when it comes to handling big data, pandas is not the only library we can resort to.

Pandas do not support task parallelization; you may work around this by invoking the multiprocessing library in Python, but it is not included out of the box. This means that calculation is limited to a single CPU core, making it somewhat slow. Pandas is not distributed; it runs on a single machine, and unless you create your framework to distribute its computations, a single machine will be the bottleneck for huge dataset computation. One other major drawback of pandas is — It doesn't scale. If your data set expands, you'll need additional RAM and, most likely, a faster processor.

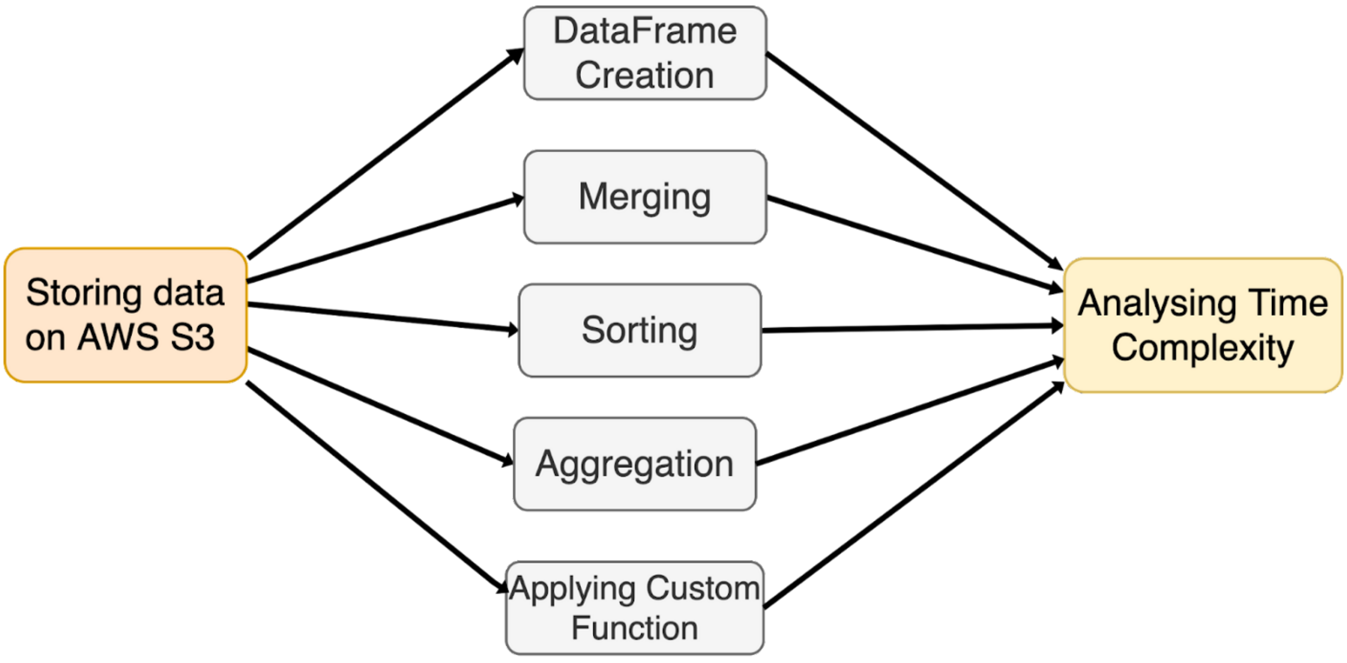
HERE COMES THE BOOM! — BIG DATA TOOLS TO THE RESCUE

Spark overcame many disadvantages of pandas, the major one was — it brought parallel processing into the game, which runs on multi-threaded programs inside the JVM and operates substantially quicker by caching data in memory across several parallel operations.

Then came many proprietary and open-source software packages like Terality and Dask. They overcame the RAM shortage and parallel computation problem of pandas. They are like pandas but on steroids! Their syntax is similar to it but they are super fast. And the surprising thing is they work independently of your machine's configuration

> *tool is suitable for a particular case.*
>
> ***TL;DR*** *— Read the Verdict* 🙂

Data Analysis Workflow

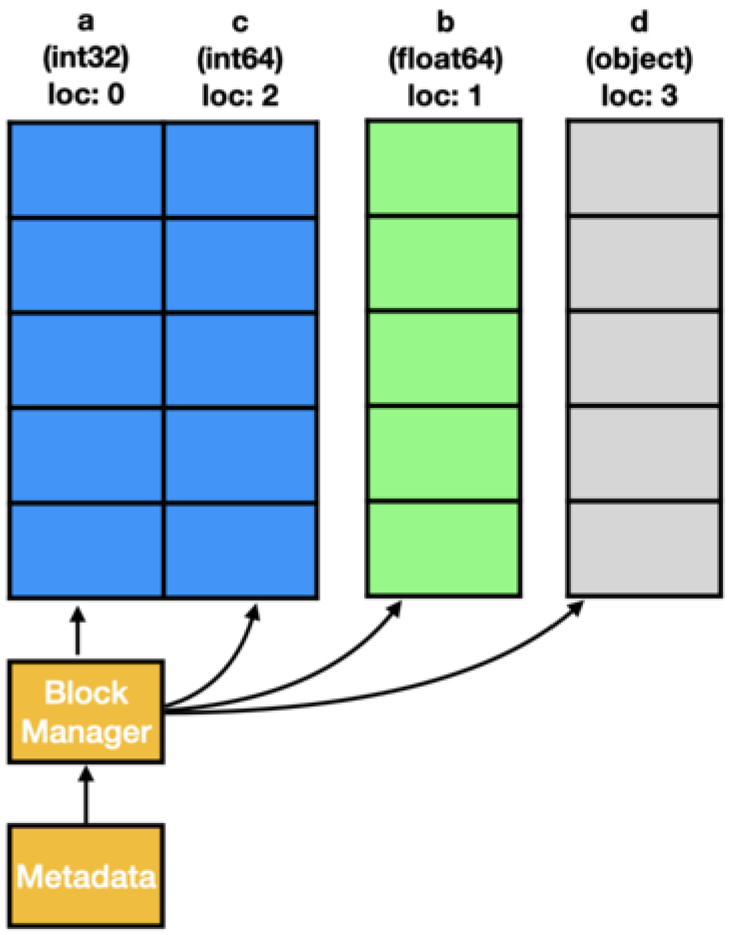


Data Analysis Workflow

To exhibit our findings we decided to use the Reddit dataset (5 GB) as our Big Data set to showcase the above-mentioned operations on the 4 data manipulation tools — Pandas, PySpark, Terality, and Dask.

Reddit has millions of users who are responsible for the expansion of the platform. With increasing users and threads, the dataset becomes an ideal candidate to test our big data analysis tools. We used the Reddit dataset which is publicly available on AWS S3 for our use-cases.

Good ol friend Pandas!



Inside Pandas DataFrame

The typical insight we can get from *pandas.DataFrame* is that it consists of metadata and each column stored as *numpy.ndarray*. The columns of the same *dtype* are kept in a single continuous memory area by the Block Manager. It monitors the actions that may appear constant in their runtime or they may be relevant to the size of your DataFrame or a subset of it.

- Pandas is a Python library for working with "relational" or "labelled" data. It provides rapid, adaptable, and expressive data structures. Its purpose is to provide a framework for conducting realistic, real-world data analysis in Python.
- Pandas is quick because it runs on top of NumPy. Numpy supports array operations that are extremely efficient. These are a set of labeled array data structures, the primary of which are Series/TimeSeries and DataFrame.
- Saving and loading pandas objects is offered by the fast and efficient

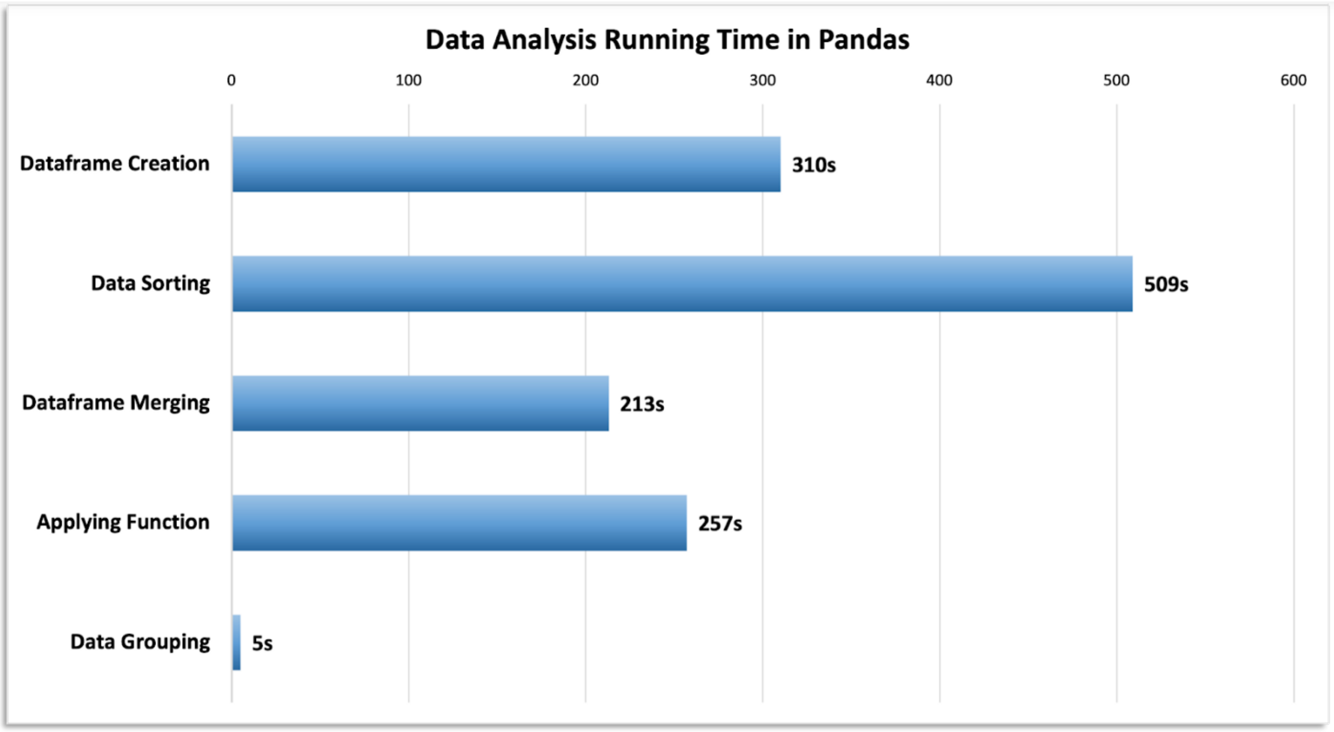
Setup

To install the latest version of pandas, run the following command

```
pip3 install pandas
```

After configuring pandas, we will perform some common data manipulation operations on Reddit Dataset using pandas and then find the time complexity for each operation.

> ***Hardware Configurations**: All the operations are performed on Apple M1 chip, 8-core CPU and 8 GB DDR4 RAM*

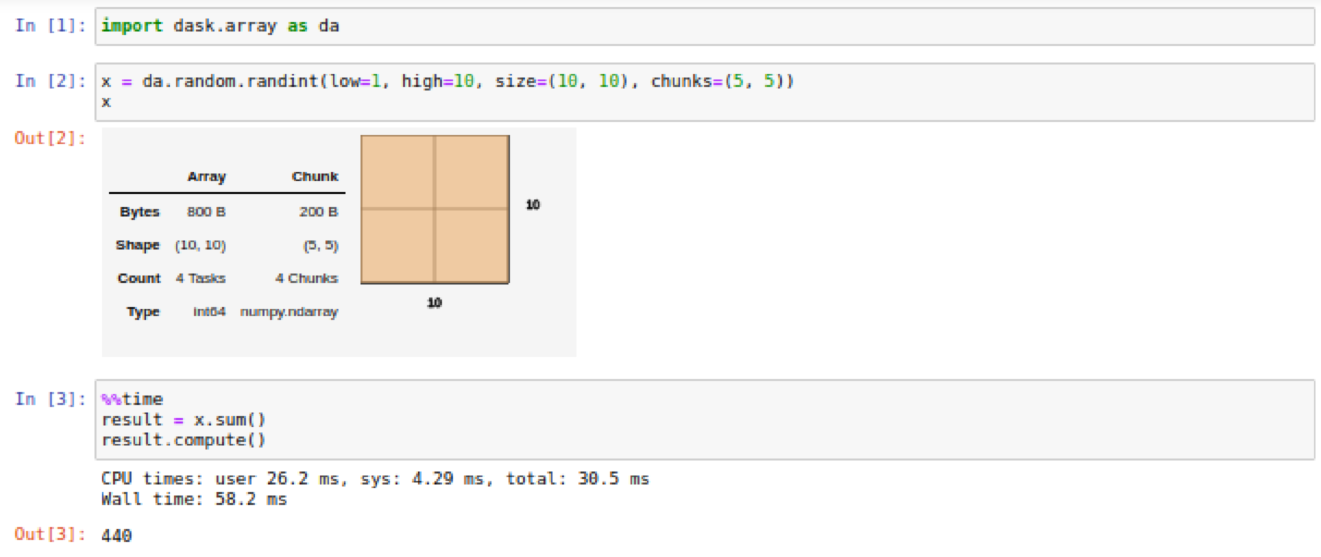


With pandas, almost all operations are taking time in the order of hundreds, which shows how inefficient it is while handling big data. One surprising result we found was that for grouping it took a surprising 5 seconds.

Just Dask it!

Dask is a Python-based open-source parallel computing framework that emphasizes virtues like familiarity, where NumPy array and Pandas DataFrame objects are parallelized. The following are the primary components that work together to allow Dask to run distributed programs with minimal effort.

- First, on chunks of data, blockwise algorithms run in parallel. Although Dask arrays and dataframes resemble NumPy and Pandas, the actual implementation of each method is reworked to run in parallel. It is not necessary for your data to fit into memory. Different pieces of data can be processed at the same time, allowing for parallelism and hence making it faster. Consider the following example of an addition operation in DASK where we first define an array of random integers.



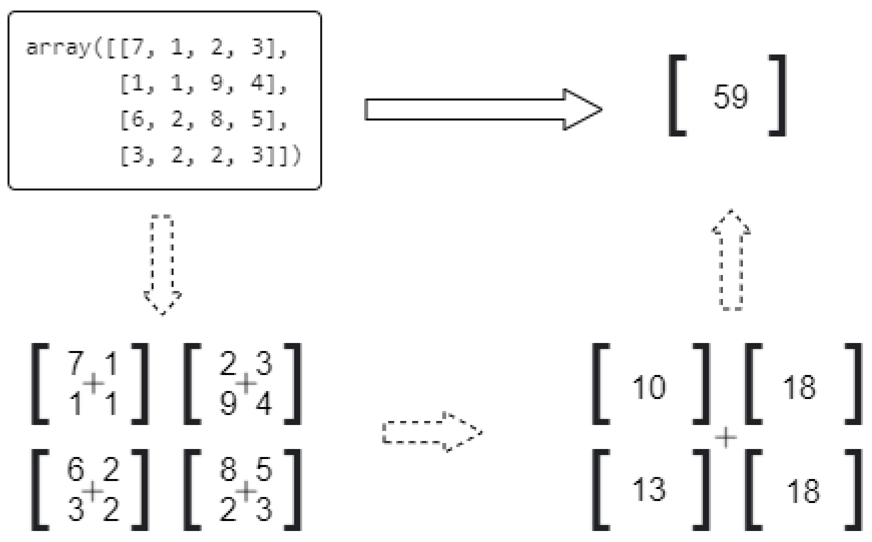
```
In [1]: import dask.array as da

In [2]: x = da.random.randint(low=1, high=10, size=(10, 10), chunks=(5, 5))
        x
Out[2]:
```

	Array	Chunk
Bytes	800 B	200 B
Shape	(10, 10)	(5, 5)
Count	4 Tasks	4 Chunks
Type	int64	numpy.ndarray

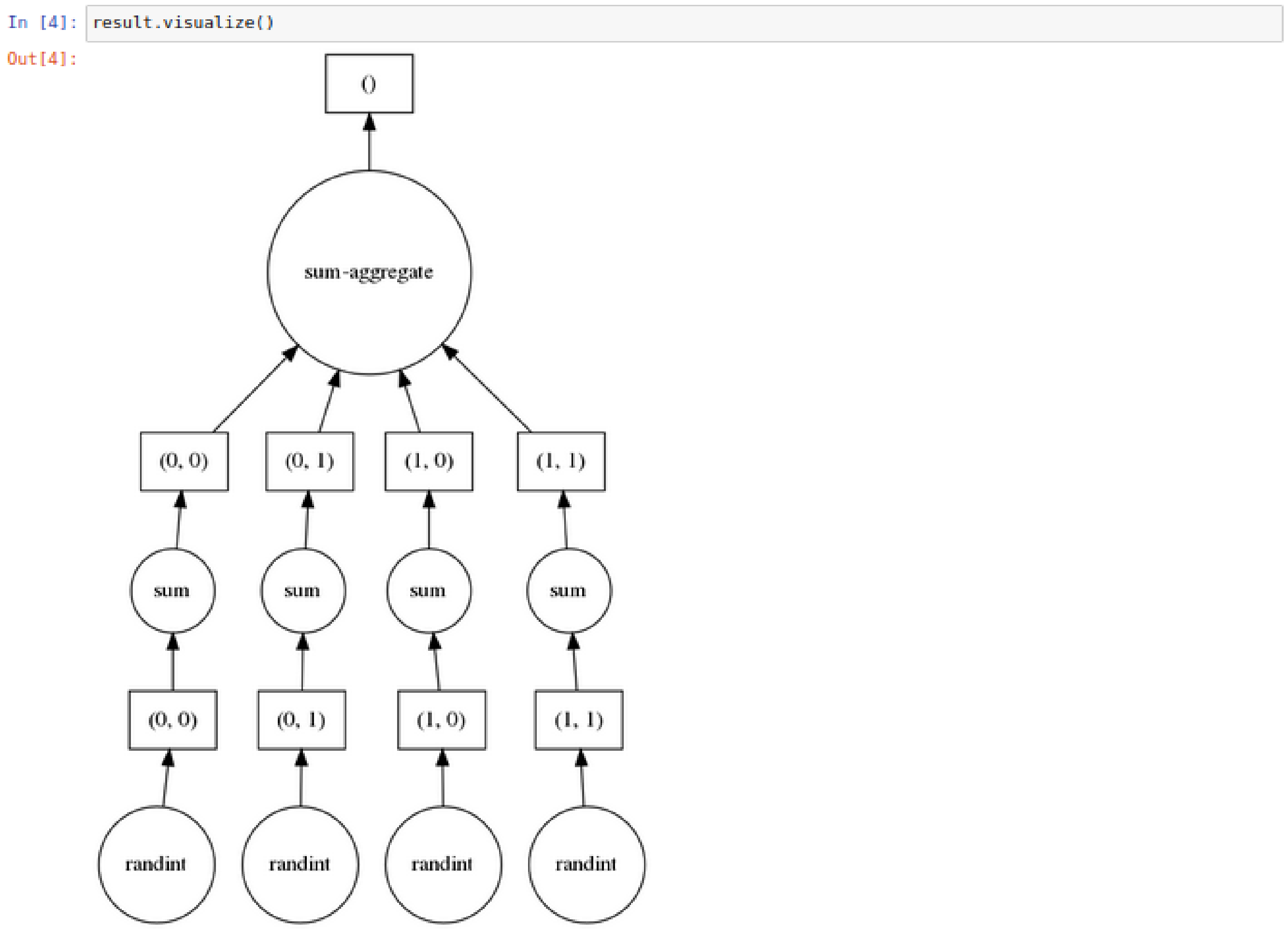
```
In [3]: %%time
        result = x.sum()
        result.compute()

        CPU times: user 26.2 ms, sys: 4.29 ms, total: 30.5 ms
        Wall time: 58.2 ms
Out[3]: 440
```

This results in 4 chunks of (5,5) array which will be processed simultaneously

- Any operation (In this instance, SUM) will be done for each chunk simultaneously to generate the final result. Because each chunk processes the sum separately and then combines the result at the end, Dask is computationally a lot faster than if the entire operation was processed sequentially by a single worker.



- Secondly, Dask, like Spark, is lazy and won't accomplish anything unless asked for. It uses python dictionaries, tuples, and functions to construct a DAG-like structure for encoding algorithms in a simple format. This structure reflects the task that the program must complete. The program is divided into a number of medium-sized tasks or computation units, each of which is often a function call on a large



```
In [4]: result.visualize()
Out[4]:
```

- And lastly, in a cluster with many nodes, which worker gets which task is decided by Scheduler. Once the computation is triggered the task graph is sent to the Dask scheduler. Parallelly, each separate task is assigned to the worker. Based on the computer specifications workers may vary. The job of the scheduler is to minimize data transfer and maximize the use of workers.

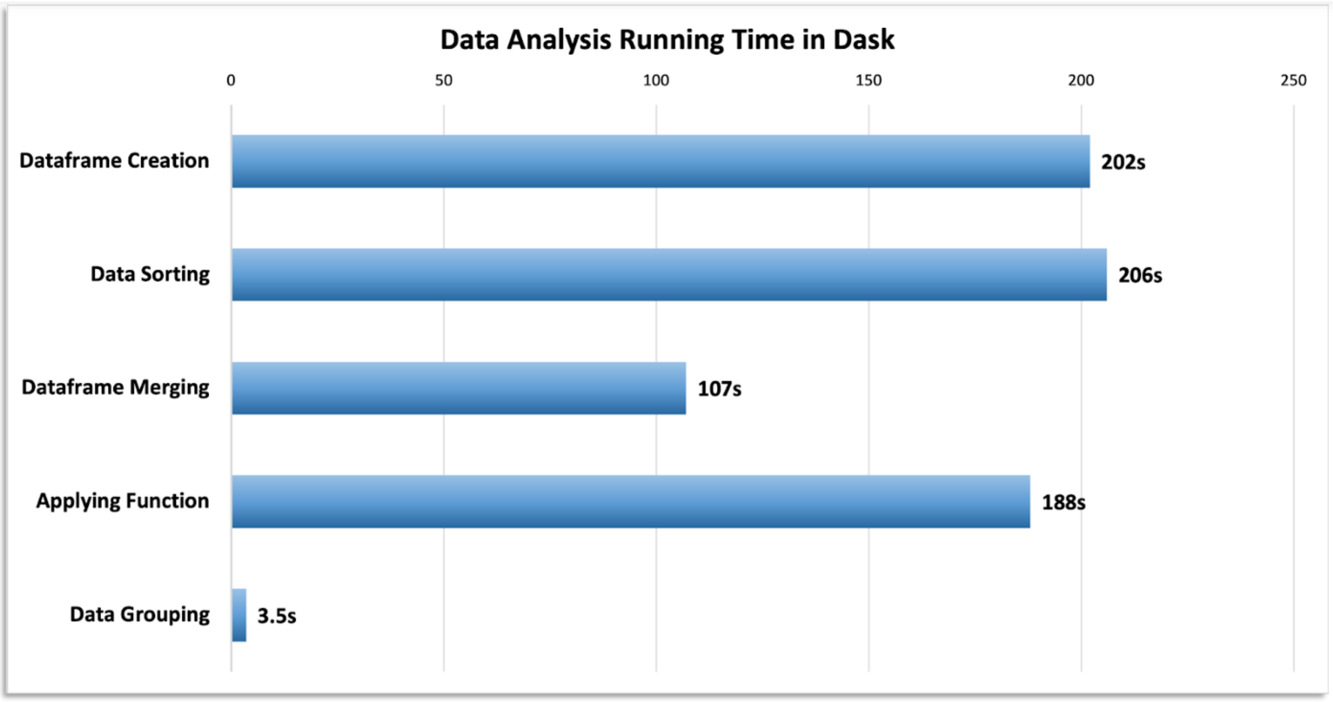
Since we have multiple workers, the **sum** happens parallelly. Which provides the performance boost that Dask provides. Furthermore, each worker only receives part of the data that is required in the computation. Since the entire dataset is not read by any worker our dataset can be as large as possible.

Setup

```
#Using pip
python -m pip install "dask[complete]"
```

```
#Using source code
git clone https://github.com/dask/dask.git
cd dask
python -m pip install
```

After configuring Dask, we will perform the same data manipulation operations on Reddit Dataset using Dask and then find the time complexity for each operation.

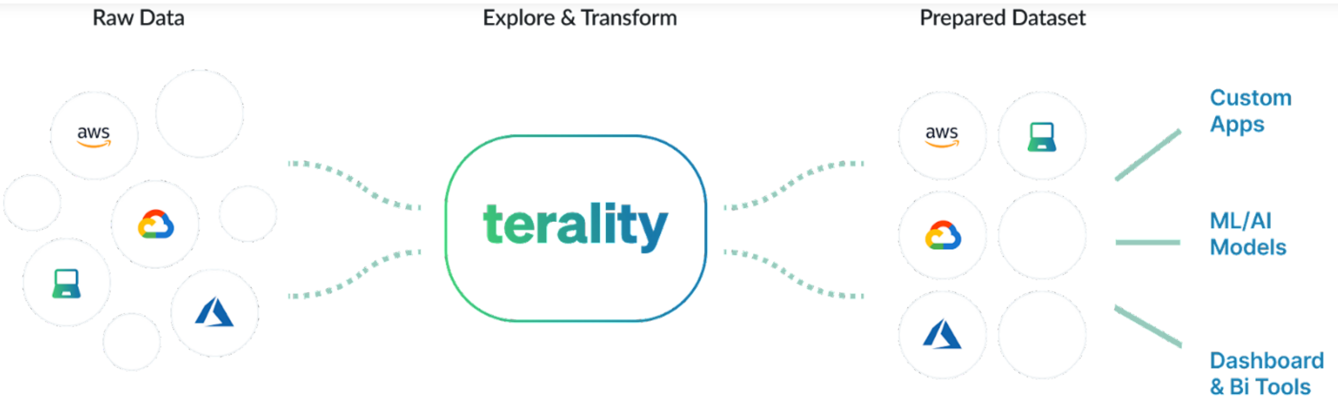


Dask performs substantially better than pandas in almost all data manipulation operations. Where the data grouping operation took almost the same time. Only had a difference of 1.5 seconds.

Terality Overview

Terality is a serverless data processing engine for data scientists and engineers. It allows data scientists and engineers to transform all of their data at lightning speed, using the same pandas' syntax, even on hundreds of gigabytes of data.

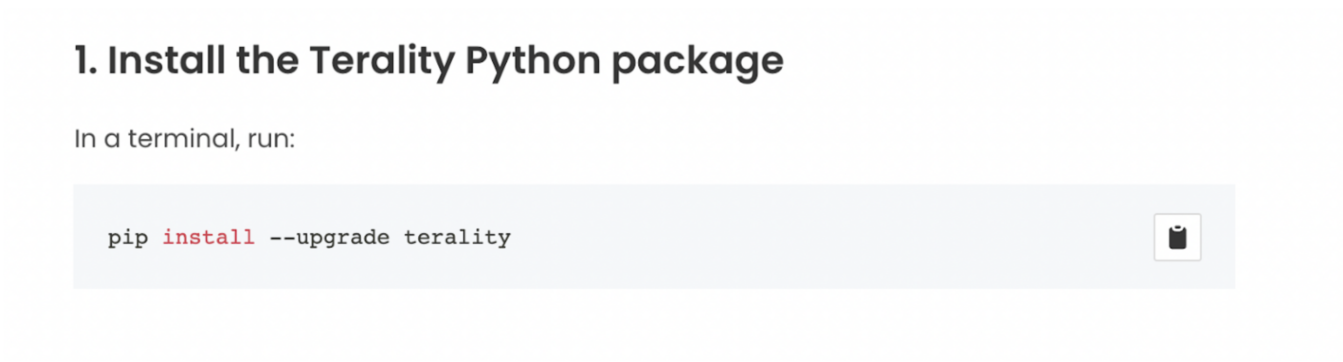
- Using an API compatible with the pandas Python library, you develop code that reads, processes, and outputs your datasets with Terality.
- You don't have to manage any infrastructure: you don't have to scale cluster resources or deploy any infrastructure. You can run your code from anywhere, such as your laptop or a virtual computer in your preferred cloud provider.
- Regardless of the size of your datasets, the Terality infrastructure auto-scales in seconds and automatically parallelizes your data processing pipelines, providing tremendous performance.
- As Terality is not open-source software, we were not able to find the underlying



Terality Workflow

Setup

Simply enter your Google or GitHub credentials during the account signup procedure. You'll be given a free 500 GB plan when you sign up, but you can always upgrade if you need more. After you've completed the registration process, you'll be taken to a dashboard page.

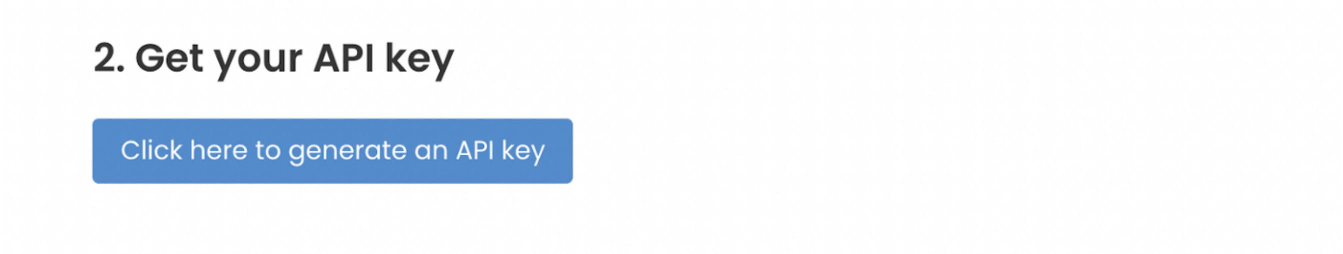


1. Install the Terality Python package

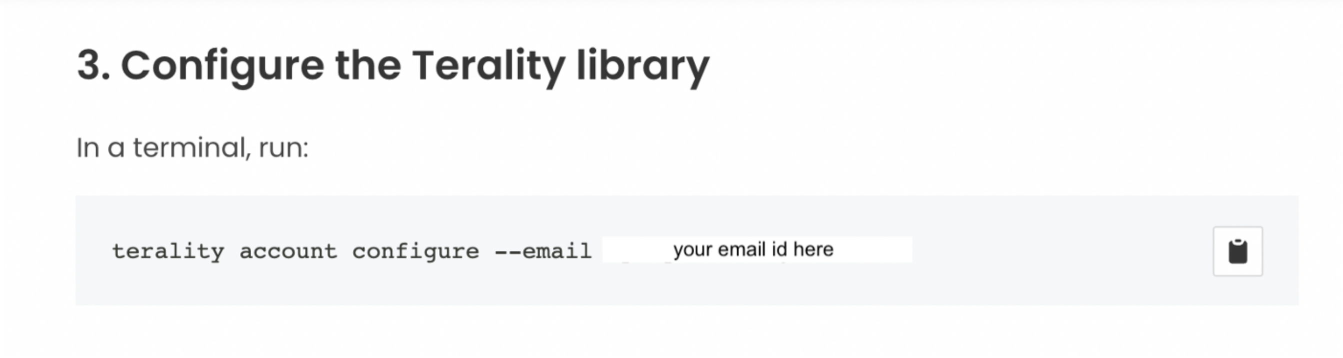
In a terminal, run:

```
pip install --upgrade terality
```

The top of the dashboard displays the remaining data processing time. The rate at which usage rises will astound you, but more on that later. Because you'll need an API key to get Terality up and running, the Quickstart section will show you how to install and set it up.



2. Get your API key

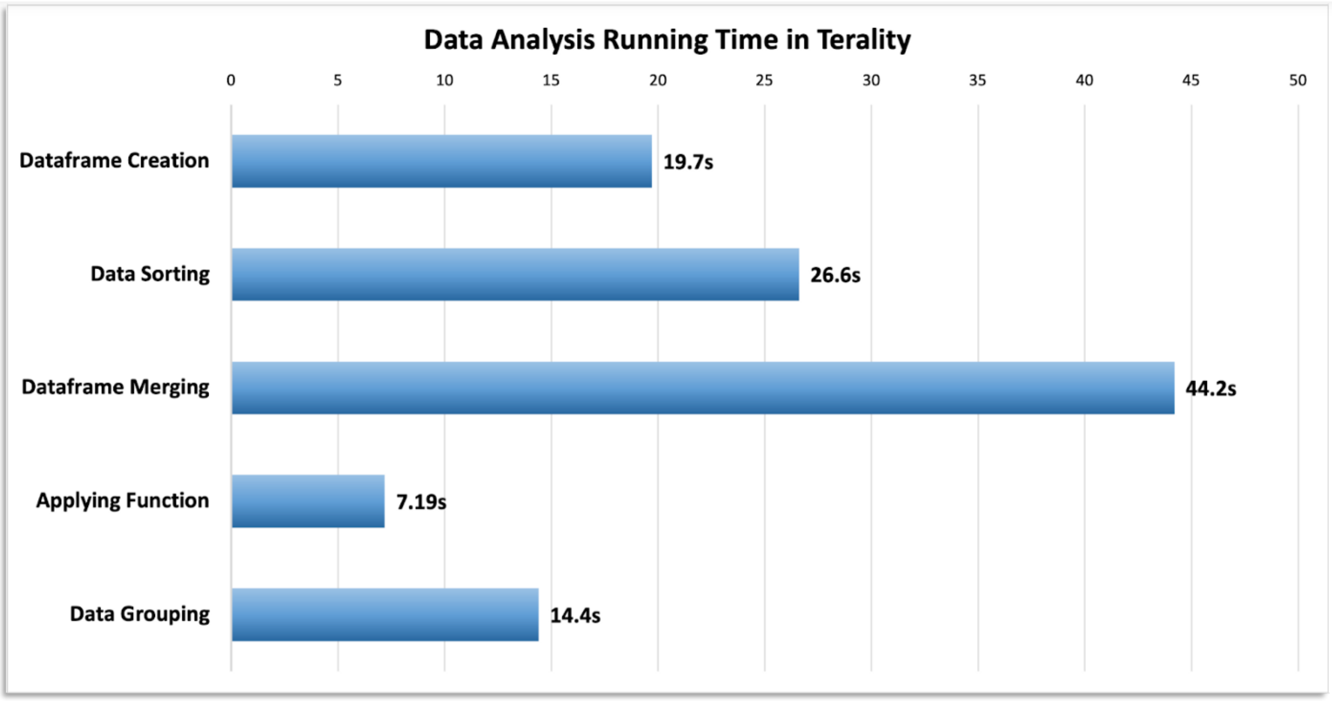


3. Configure the Terality library

In a terminal, run:

```
terality account configure --email your email id here
```

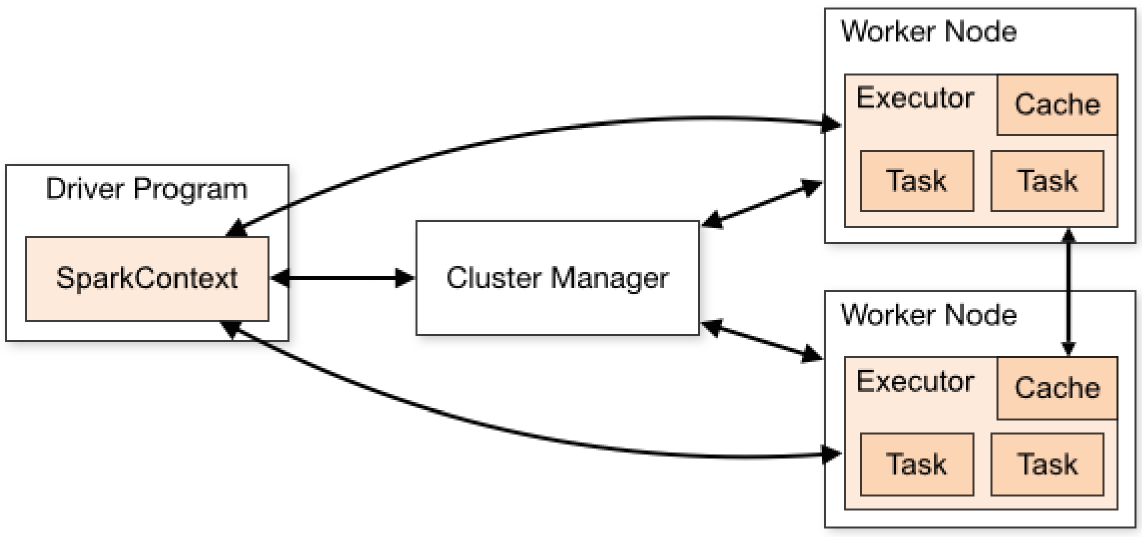
After configuring Terality, we will perform the same data manipulation operations on Reddit Dataset using Terality and then find the time complexity for each operation.



Terality showed improvements in almost all data manipulation operations in comparison with pandas and dask. Where the merging operation took the maximum amount of time and applying a custom function showed the minimum amount of time.

> ***NOTE:*** *The speed of Terality is determined by your Internet connection, not your CPU. Reading the Parquet file from memory would necessitate transferring 5GB of data to the engine, which would be inefficient.*

PySpark — Say hello to stream data



The Apache Spark framework employs a master-slave design, with a driver serving as the master node and a large number of executors serving as the cluster's worker nodes.

According to the one task per partition concept, the cluster manager or resource manager entity of Spark delegates the tasks of running the Spark jobs to the worker nodes. Various iterations techniques are performed on the data repeatedly to cache the datasets across iterations. Each job performs its unit of operations on the dataset within its partition, resulting in a newly partitioned dataset. These findings are returned to the main driver application for further processing or storage.

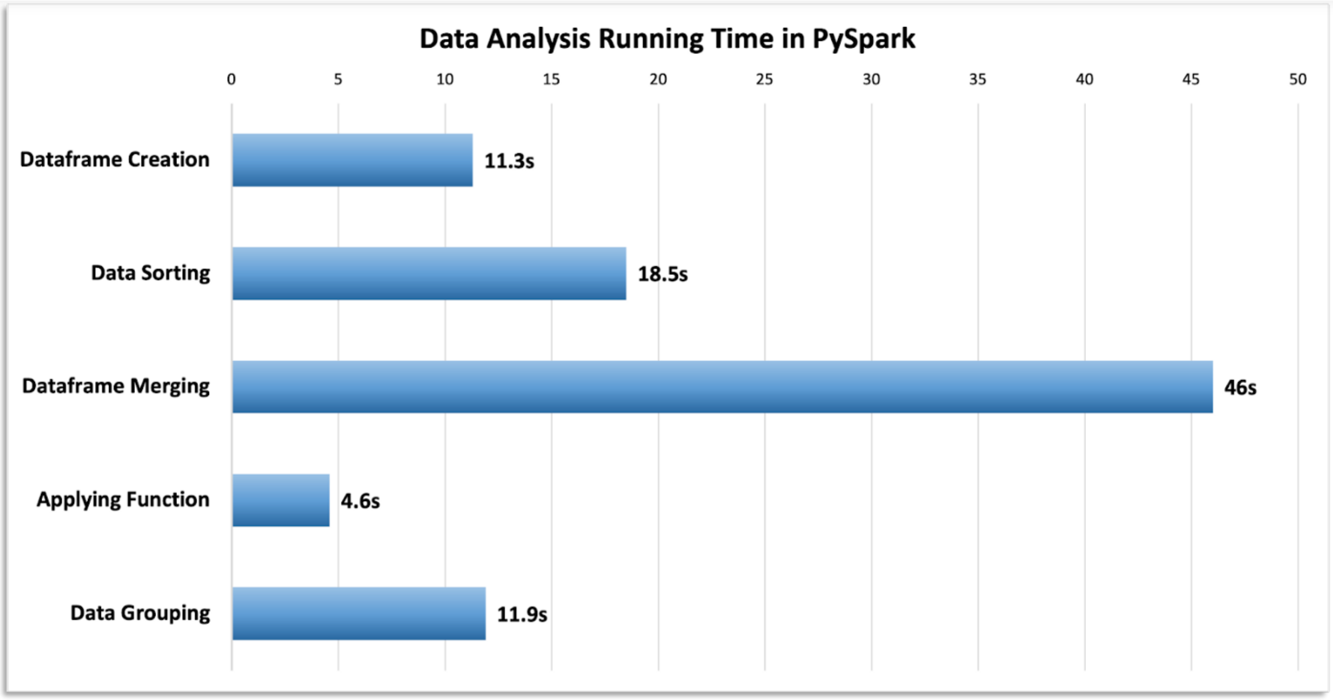
- PySpark is a Python interface to Apache Spark. It includes the PySpark shell for interactively examining data in a distributed environment, as well as the ability to develop Spark applications using Python APIs.
- It allows you to interact with Resilient Distributed Datasets (RDDs) in Apache Spark. Using the Py4j library, we were able to do this. PySpark LogoPy4J is a popular package that is built into PySpark that allows Python to interact with JVM objects dynamically.
- As we know PySpark is Python-based hence its APIs are very easy and simple to apply. Plus Python's code readability, maintenance, and familiarity are considerably superior.

Setup

```
#Install Prerequisite

/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install

/bin/bash -c $(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh
)

#Install Java and Scala
$ brew cask install java
$ brew install scala

#Install Spark
```

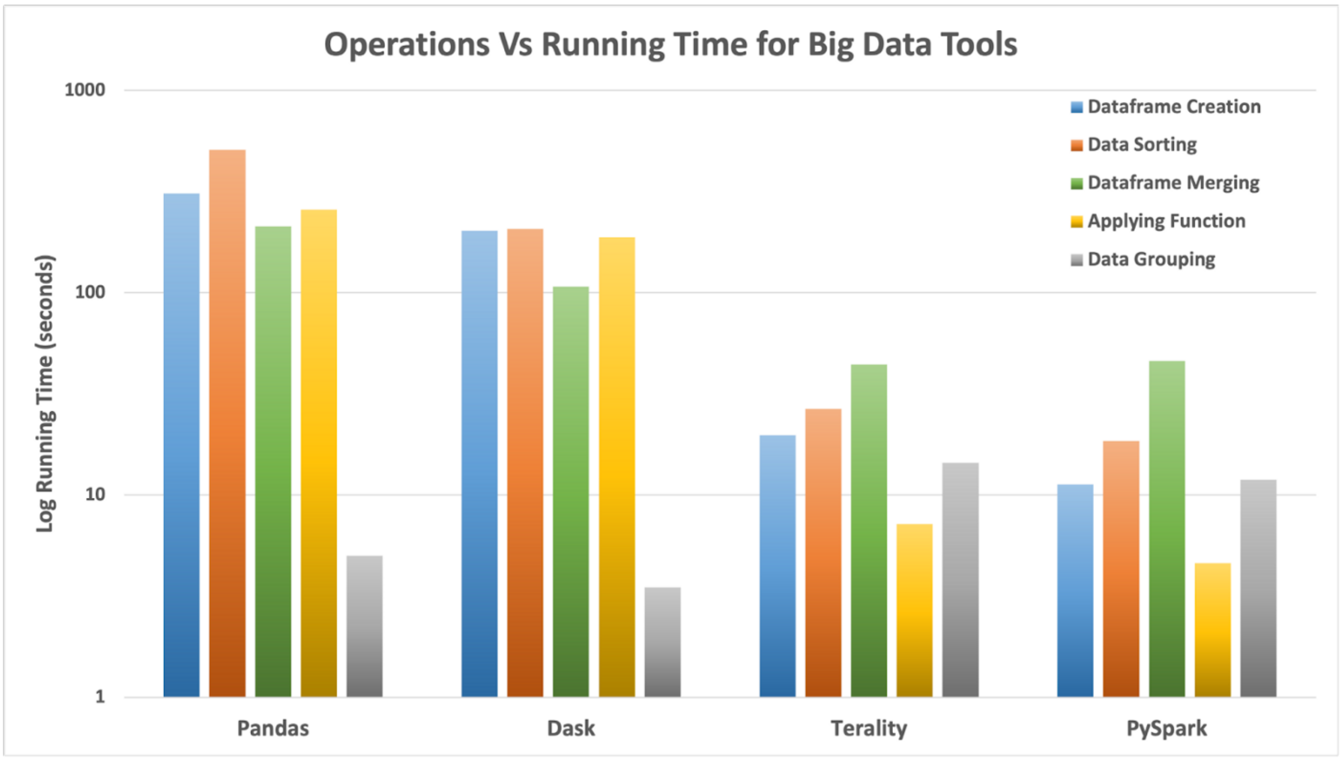
> *Declare variable path for pyspark using export command and can verify by running* ***pyspark*** *in the terminal and can access spark user interface on this URL:* http://localhost:4040/jobs/

After configuring PySpark, we will perform the same data manipulation operations on Reddit Dataset using PySpark and then find the time complexity for each operation.



Data manipulation in PySpark was way better than its counterparts. Only Terality showed comparable performance in each aspect. The most inefficient operation was dataframe merging.

Conclusion



technologies. Terality, on the other hand, trails behind PySpark in every respect except merging, when Terality outperforms PySpark by only 2 seconds.

The Verdict

> *Terality makes it simple to use by supporting the pandas syntax, which improves parallelization and speeds it up to 10–100 times quicker than pandas. Dask additionally has a processing benefit over pandas on small to medium-sized datasets because it supports the pandas syntax asynchronously. Terality's serverless design, on the other hand, has an advantage over dask's local infrastructure. On the contrary, PySpark has a computational advantage in all parts of data processing; nevertheless, because of the semi-managed infrastructure, where the user may need to scale the servers based on the data size, Terality would be chosen because the auto-scaling takes care of surpassing data restrictions. Hence, choosing a preferred big data tool is not an easy thing — Hopefully, with this blog, you will get a better idea.*

Get unlimited access Open in app

References

- https://www.terality.com/
- https://github.com/dask/dask
- https://docs.dask.org/en/stable/
- https://spark.apache.org/docs/latest/api/python/
- https://pandas.pydata.org/docs/user_guide/index.html
- https://uwekorn.com/2020/05/24/the-one-pandas-internal.html

D