Published in Towards Data Science

Paras Varshney Follow

Oct 5, 2020 · 5 min read · · Listen

Save

MAKING SENSE OF BIG DATA, DATA SCIENCE, MACHINE LEARNING

DASK: A Guide to Process Large Datasets using Parallelization

A simple solution for data analytics for big data parallelizing computation in Numpy, Pandas, and Scikit-Learn Frameworks

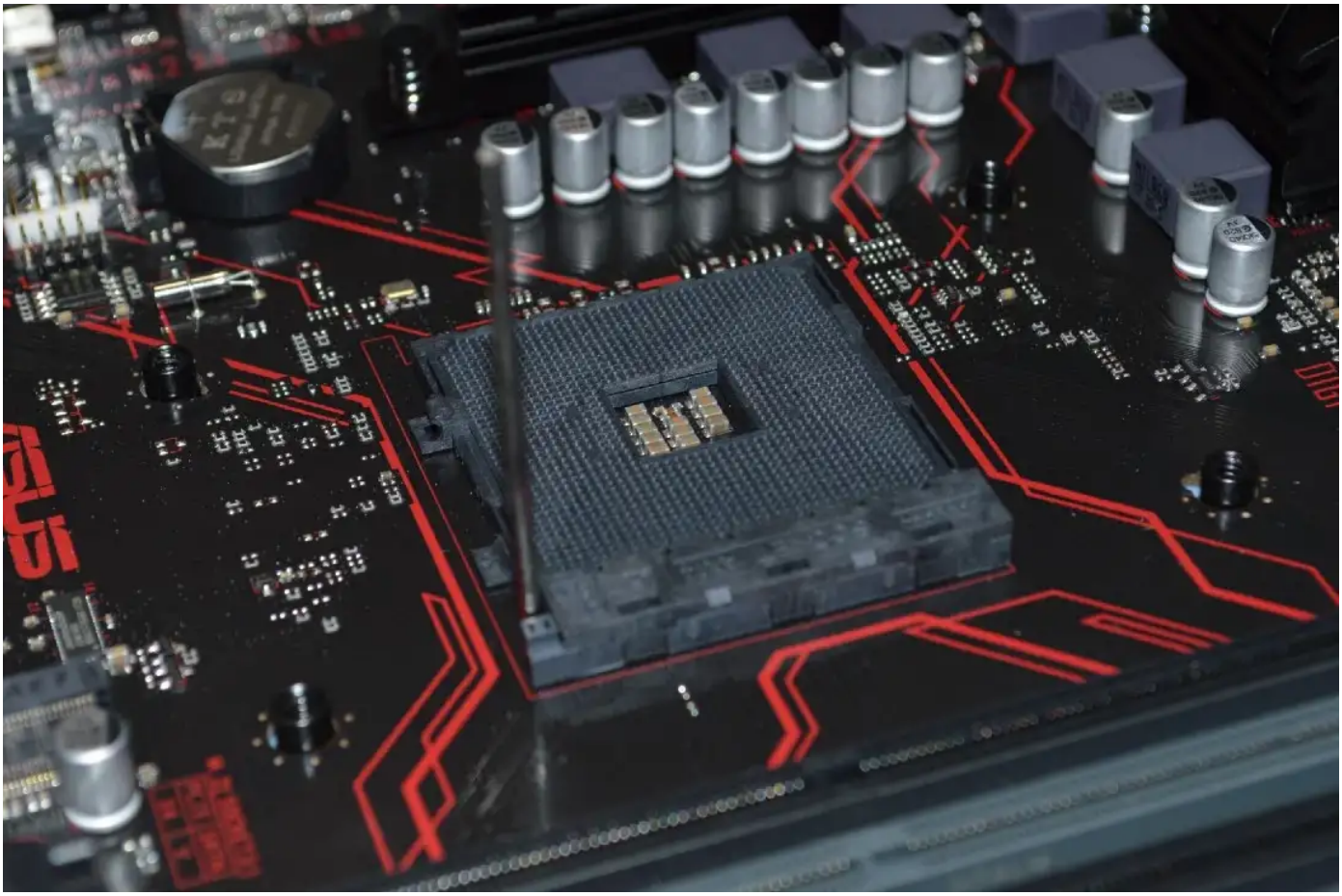


Photo by Thomas Jensen on Unsplash

Introduction

If you are dealing with a large amount of data and you are worried that Pandas' data frame is unable to load it or NumPy arrays get stuck in between and you even need a much better and parallelized solution for your data processing and training machine learning models then dask open up a solution to this problem. Before diving into that, let's see what actually is dask?

Before diving-in deep, have you ever heard about Lazy-Loading? Check out how Vaex is dominating the market of loading huge datasets.

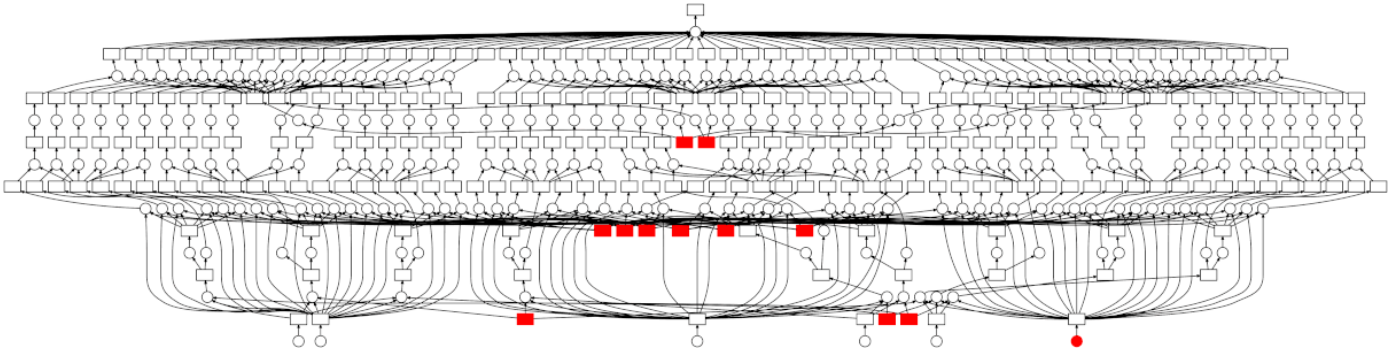
Now, Load huge datasets within a second ⚡ using lazy computation 😴 in Python?

Tired of loading datasets with pandas... Learn how Vaex helps in loading a huge amount of data within seconds with...

towardsdatascience.com

What is dask?

Dask is an extremely efficient open-source project that uses existing Python Apis and knowledge structures that makes it straightforward to modify between Numpy, Pandas, Scikit-learn into their Dask-powered equivalents. Also, Dask's schedulers scale to thousand-node clusters and its algorithms are tested on a number of the most important supercomputers within the world.



Source: Scale up to clusters using Dask Parallelization

Installation

Does quality comes pre-installed inside your Anaconda but for pip you can get the complete one using this command:

Conda installation for Dask:

```
!conda install dask
```

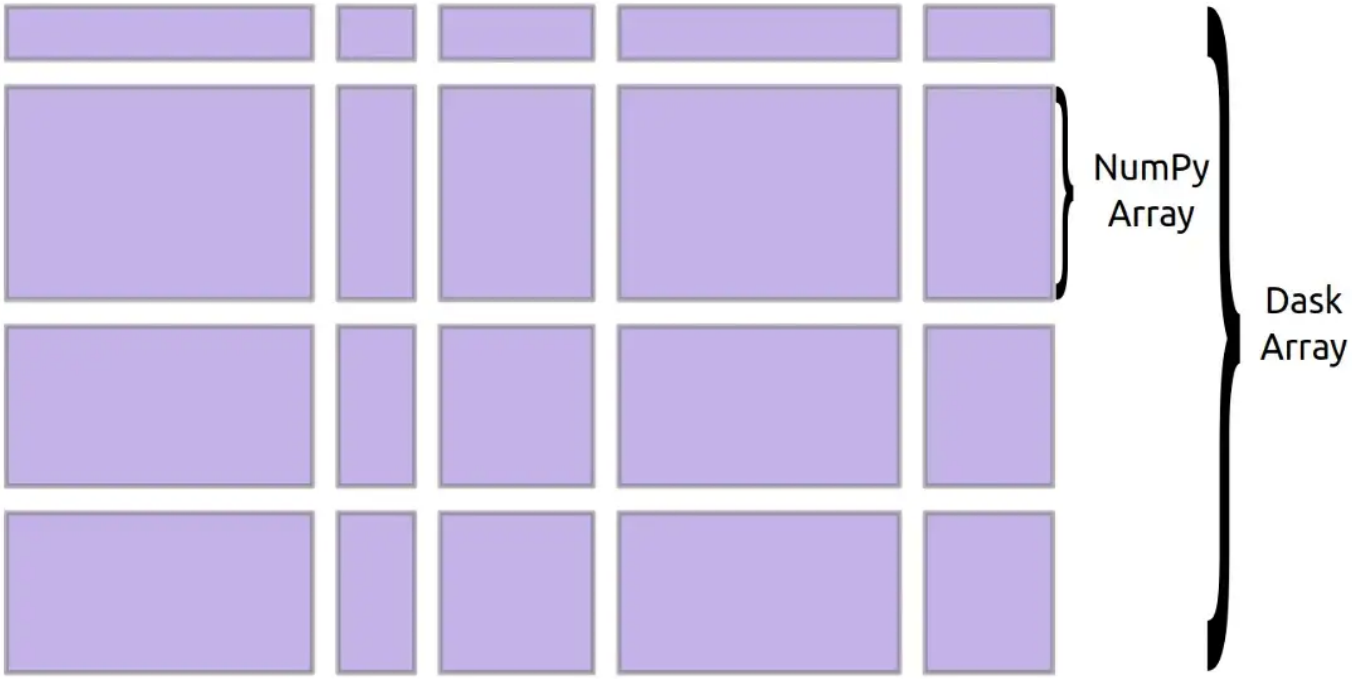
pip installation for Dask:

```
!pip install "dask[complete]"
```

What does Dask do?

Dask helps to parallelize Arrays, DataFrames, and Machine Learning for dealing with a large amount of data as:

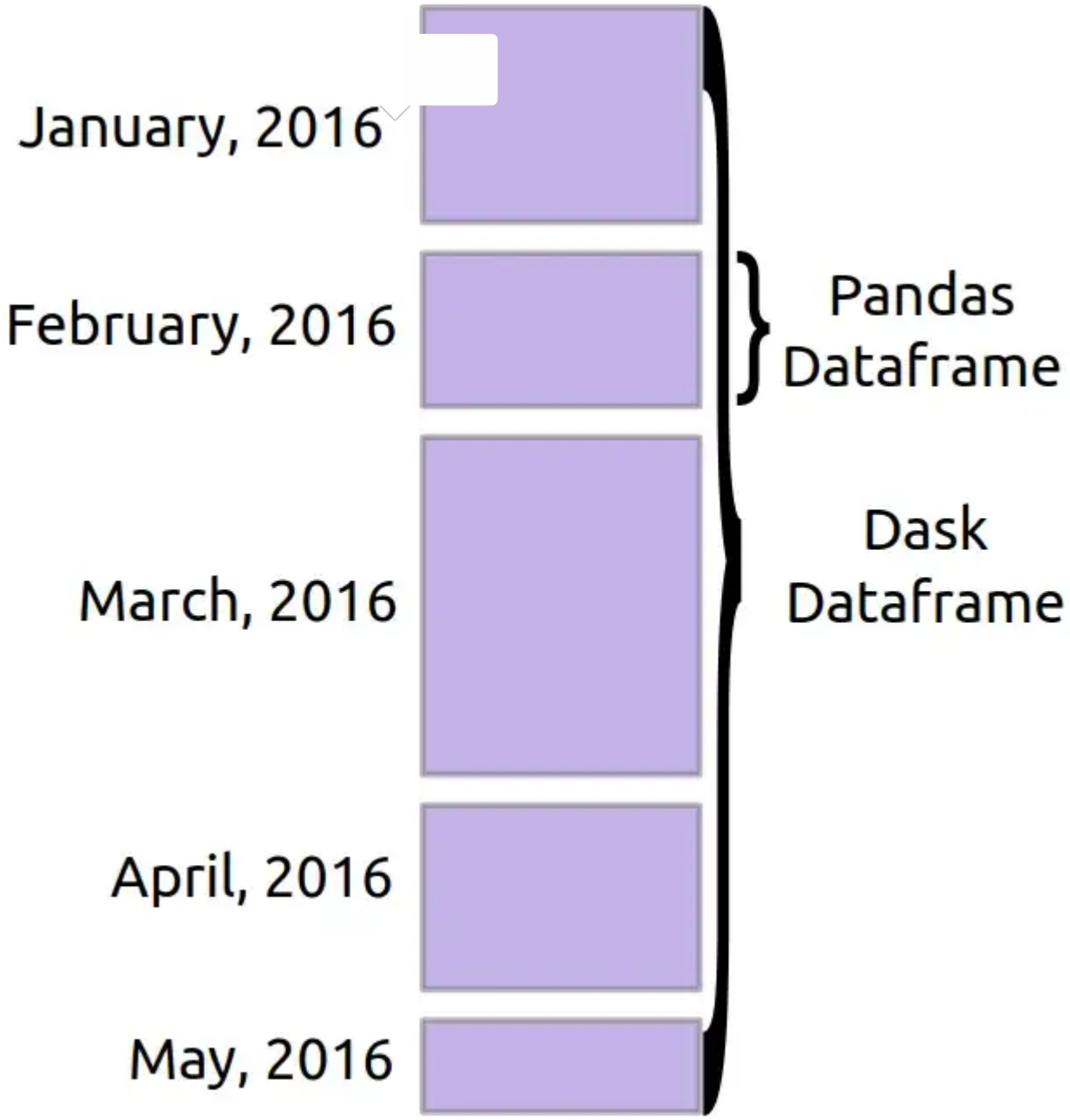
Arrays: Parallelized Numpy




```
# Arrays implement the Numpy API

import dask.array as da
x = da.random.random(size=(10000, 10000), chunks=(1000, 1000))
x + x.T - x.mean(axis=0)
```

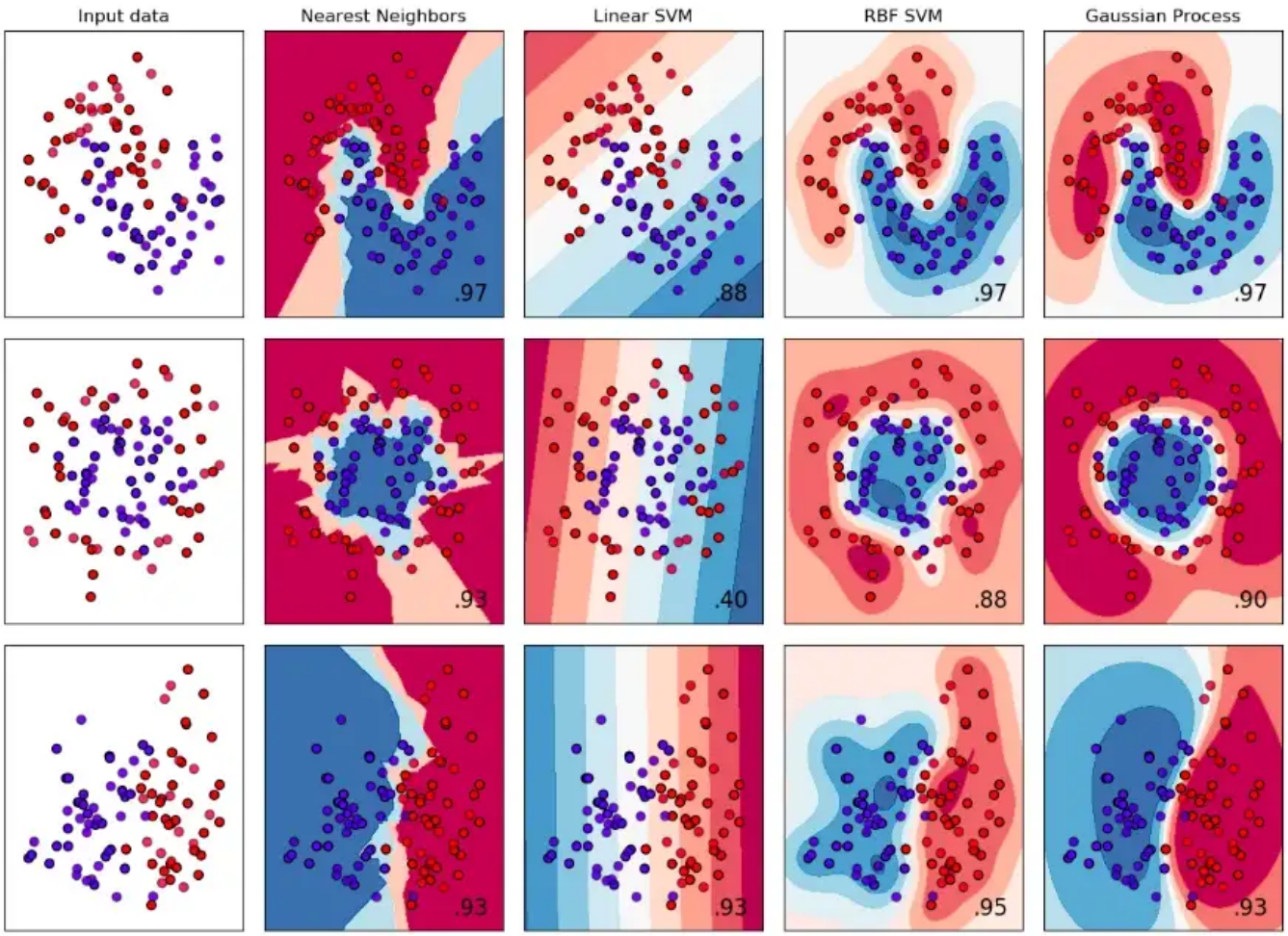
DataFrame: Parallelized Pandas




```
# Dataframes implement the P

import dask.dataframe as dd
df = dd.read_csv('financial_dataset.csv')
df.groupby(df.amount).balance.sum()
```

Machine Learning: Parallelized Scikit-Learn




```
# Dask-ML implements the Scikit-Learn API

from dask_ml.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train, test)
```

DataFrame in Dask

Most of the Dask API is very similar to the Pandas API so you can directly use the data frames of the Pandas in Dusk with a very similar command. To generate a discrete data frame you can just simply call the `**read_csv()**` method in the same way you used to call in Pandas or can easily convert a Pandas DataFrame into a Dask DataFrame.

```
import dask.dataframe as ddf
dd = ddf.from_pandas(df, npartitions=N)
```

Benchmarking DataFrame: Pandas vs Dask

For the following benchmarking the machine used had a standard 4-core processor which stays standard for testing both frameworks.

I have done a very simple yet interesting benchmark to show how fast is Dask DataFrame as compared to a traditional Pandas DataFrame for reading a dataset from a .csv file having 5 million records.

```
In [35]: import time
         import pandas as pd
         import dask.dataframe as ddf

In [36]: start = time.time()

         df = pd.read_csv("sample_data.csv")
         print("Time Taken: "+str(time.time()-start))

         Time Taken: 6.272840738296509

In [37]: start = time.time()

         df = ddf.read_csv("sample_data.csv")
         print("Time Taken: "+str(time.time()-start))

         Time Taken: 0.07356977462768555
```

Benchmarking Pandas vs Dask for reading CSV DataFrame

Results: To read a **5M** data file of size over **600MB** Pandas DataFrame took around **6.2 seconds** whereas the same task is performed by Dask DataFrame in much much less than **a second** time due to its impressive parallelization capabilities.

Note: This test was done on a small dataset, but as soon as the size of data increases, this time difference in reading data gets exponentially high.

You can use the code below to alter the benchmarking for a bigger dataset.

```python
import time
import pandas as pd
import dask.dataframe as ddf


# benchmark Pandas Dateframe for time taken to read a csv file with 5M records.
start = time.time()
df = pd.read_csv("sample_data.csv")
print("Pandas Time Taken: "+str(time.time()-start))

# benchmark Dask Dateframe for time taken to read a csv file with 5M records.
start = time.time()
df = ddf.read_csv("sample_data.csv")
print("Dask Time Taken: "+str(time.time()-start))
```

dask_benchmarking.py hosted with ❤ by GitHub view raw

Benchmarking Array: Numpy vs Dask

In this Benchmark, I generated a **1Trillion** sized array of random numbers using both Numpy Array as well as Dask Array.

```
In [21]: import numpy as np
         import dask.array as da

In [22]: start = time.time()

         np.random.randint(0,100000,(1000000000))
         print("Time Taken: "+str(time.time()-start))

         Time Taken: 7.888777732849121

In [23]: start = time.time()

         da.random.random((0, 100000), chunks=(1000000000))
         print("Time Taken: "+str(time.time()-start))

         Time Taken: 0.0008273124694824219
```

Benchmarking Pandas vs Dask for creating an array

Result: As expected the results were pretty obvious as Numpy Array took a bit less than **8 seconds** to compute whereas the Dask Array took **negligible** time!

You can try out the same benchmark using the code below

```
1  import time
2  import numpy as np
3  import dask.array as da
4
5  # Genrate a random numbers array of size 1 Trillion for Numpy Array.
6  start = time.time()
7  np.random.randint(0,100000,(1000000000))
8  print("Time Taken: "+str(time.time()-start))
9
10 # Genrate a random numbers array of size 1 Trillion for Dask Array.
11 start = time.time()
12 da.random.random((0, 100000), chunks=(1000000000))
13 print("Time Taken: "+str(time.time()-start))
```

dask_benchmark_Numpy_Array.py hosted with ❤ by GitHub view raw

Key Take-Aways

Dask helps in doing data analysis faster because it parallelizes the existing frameworks like Pandas, Numpy, Scikit-Learn, and process data parallelly using the full potential of your machine's CPU. You can try experimenting with the amazing features of Dask here.

The combination of Lazy Loading with Parallel Processing is really a deadly combination and helps you to utilize the full potential of your system whenever required, to know more you can read this article about Vaex.

Further Reading for Data Scientists:

How to Evaluate Machine Learning Model Performance in Python?

A Practical Approach to Compute the Model's Performance and Implementation in Python covering all Mathematical...

medium.com

How to learn Data Science from Beginners to Masters in just 1 year (my personal experience)

The complete compilation of my checklist to learn Data Science for a Beginner to a master is just one year with time...

towardsdatascience.com

Thank You!

Big Data Analytics Data Science Machine Learning Making Sense Of Big Data

Thanks to Towards AI Editorial Team
