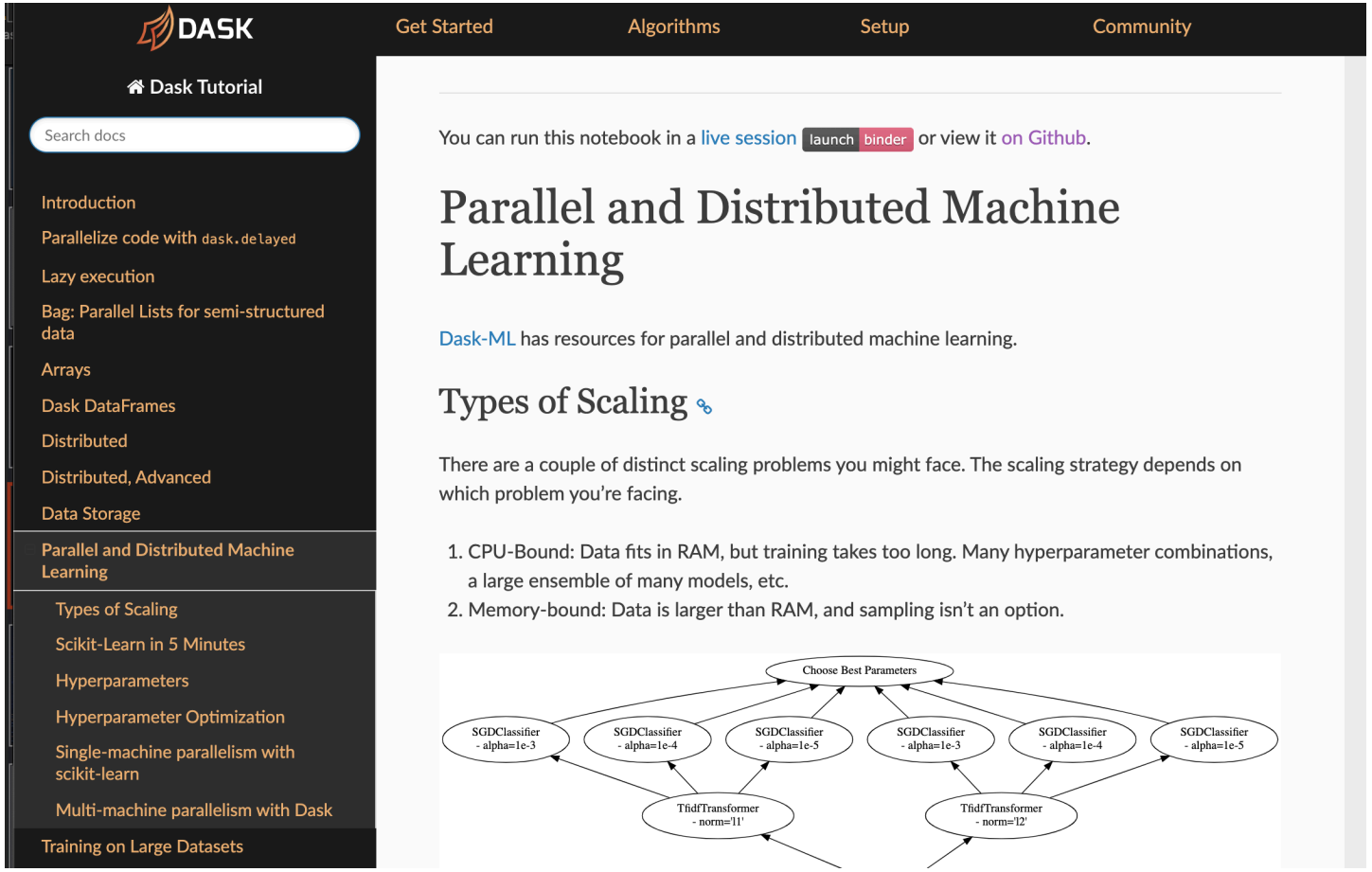

Dasking in the Sun
Applying Parallel Operations to your Pandas Transformations

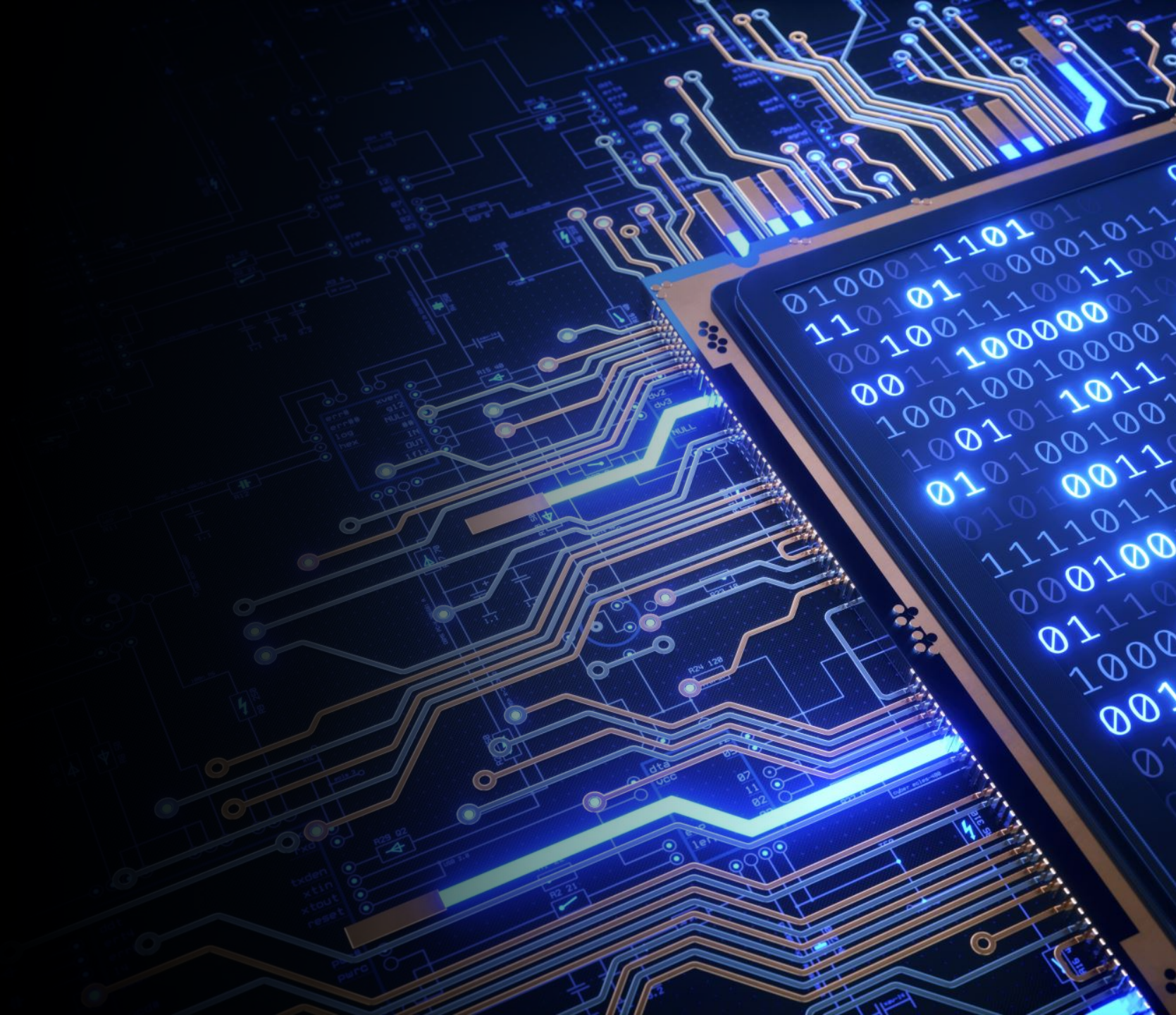
What do you do when your data become too large? Is it CPU-bound? Or memory-bound?

Dask docs provide guidance



- ***RAM:*** Your computer's short-term data storage. Can be accessed quickly.
- ***Cores:*** Modern CPUs offer cores and hyper-threading. A dual-core has 2 CPUs, quad-core has 4 CPUs, octo-core has 8, and so on.

- Processes tend to be memory-bound or CPU-bound, meaning that either the data are very large and you might run out of memory, or the number of computations is so great that you might run out of CPU.

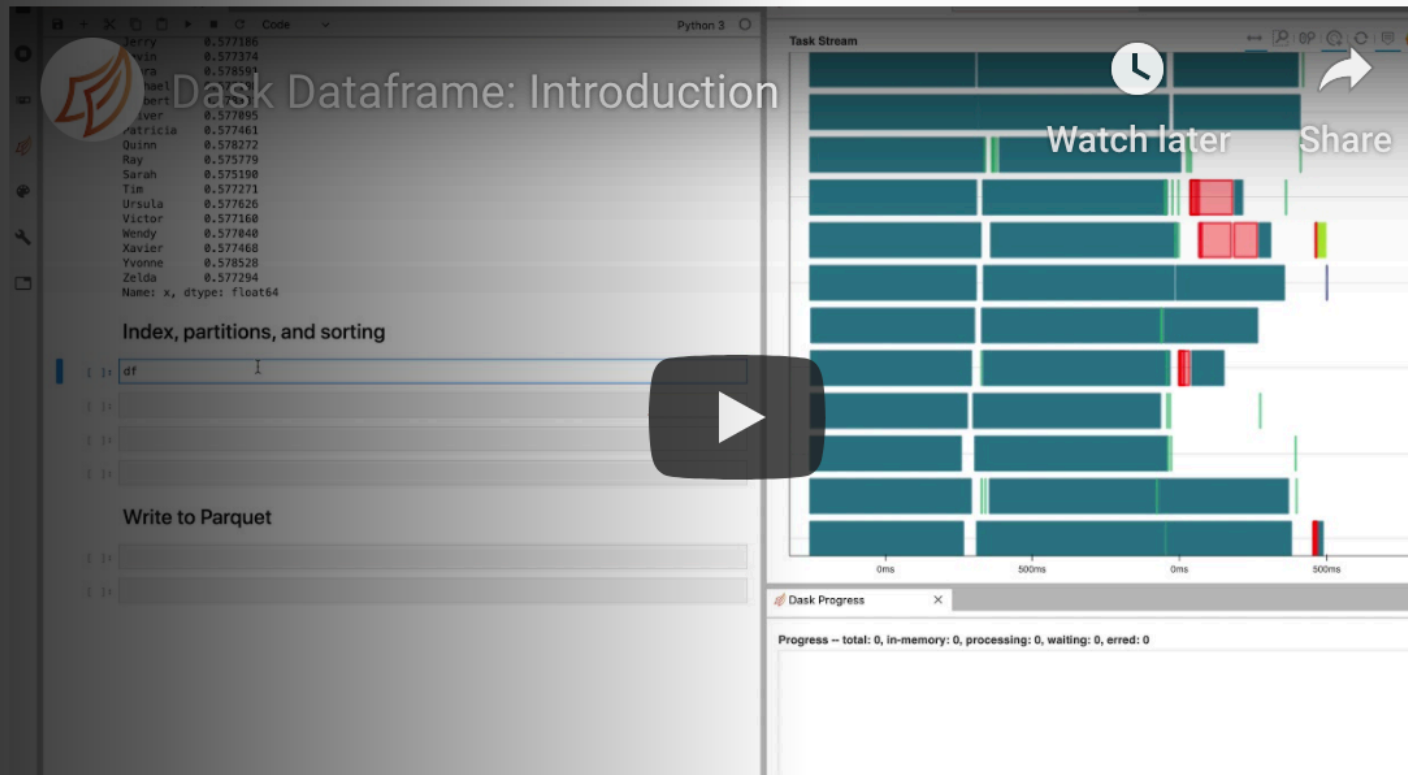


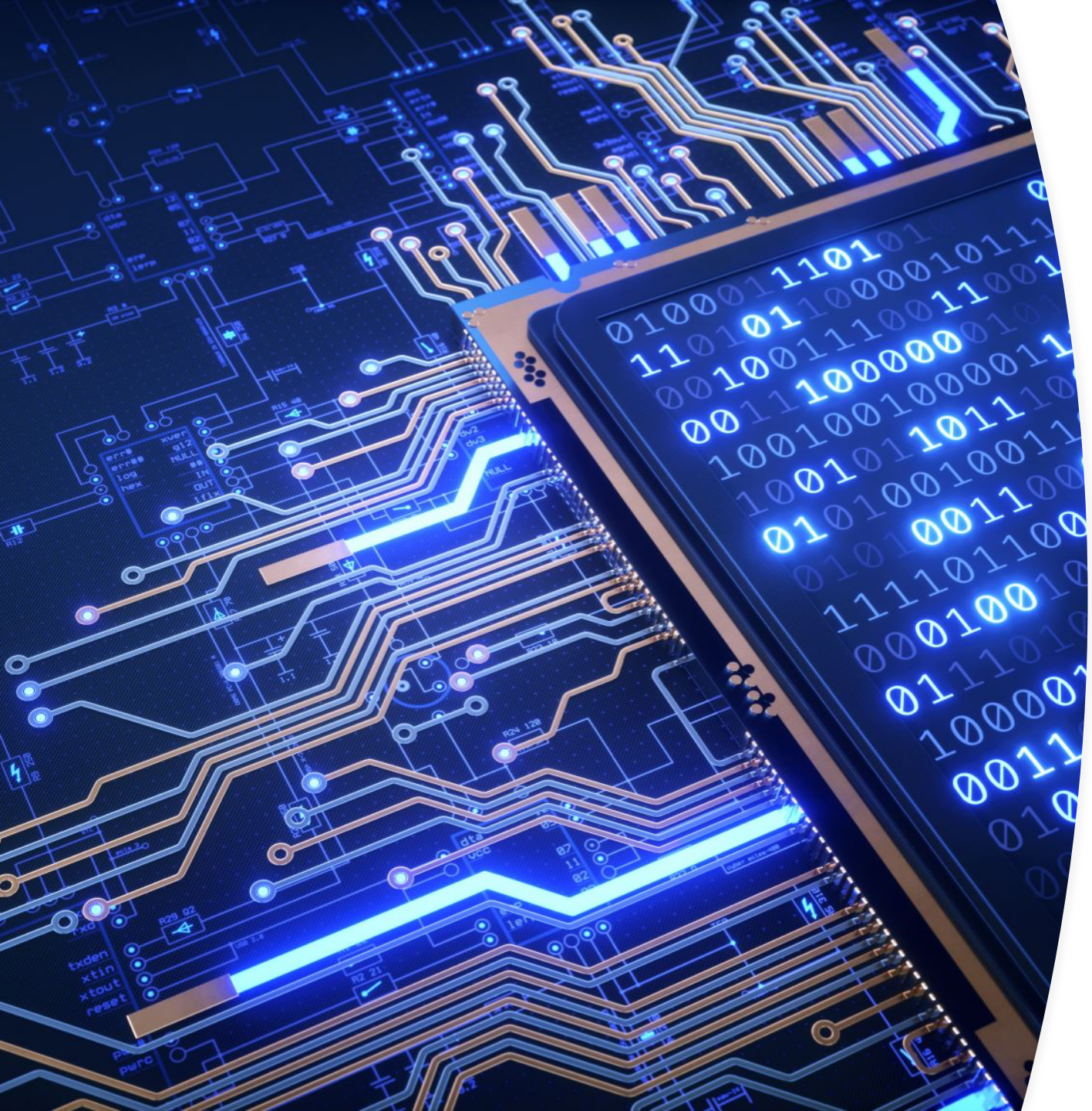
- Today's code was run on:
  - Python virtualenv created with venv
  - Requirements are in requirements.txt
  - Python 3.7.6
  - MacOS

DASK

DataFrame

A Dask DataFrame is a large parallel DataFrame composed of many smaller Pandas Dat
split along the index. These Pandas DataFrames may live on disk for larger-than-memor
on a single machine, or on many different machines in a cluster. One Dask DataFrame c
triggers many operations on the constituent Pandas DataFrames.





- The basic concept of Dask is to split data/computation up across a computer's CPU.

- Dask is installable:

```
$conda install dask
```

```
$pip install dask
```

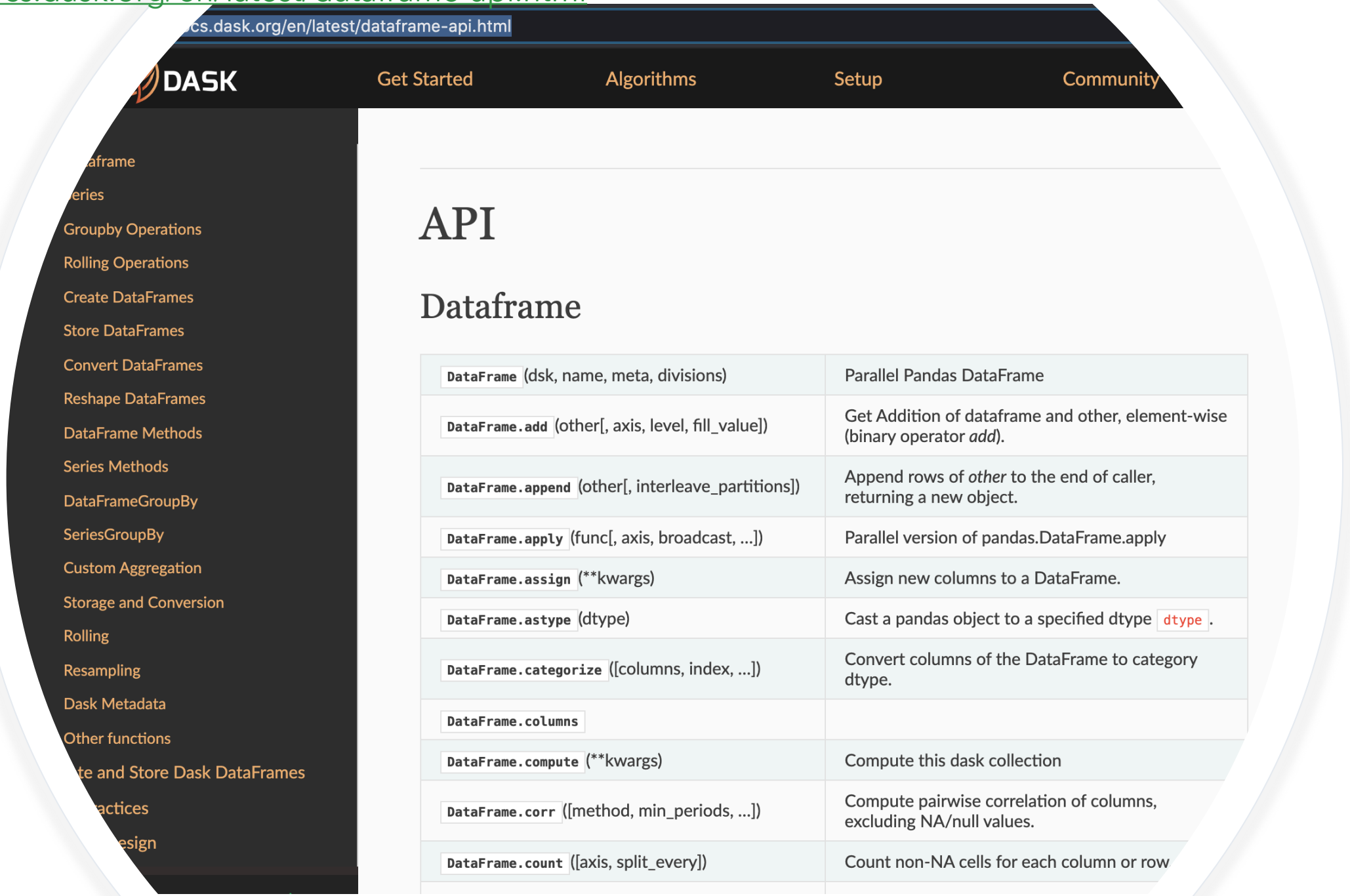
- Or from source:

```
$git clone https://github.com/dask/dask.git
$cd dask
$python setup.py install
```

Dask

- Can scale from a laptop to a cluster
- Has the concept of arrays and dataframes
- Has several ways of distributing tasks and scheduling them, including dask.delayed
- Has dask-ml, parallelized machine learning that works alongside scikit-learn (scikit-learn already has joblib, but Dask extends this package to clusters)

https://docs.dask.org/en/latest/dataframe-api.html



- Multiprocess: https://docs.python.org/3/library/multiprocessing.html
- Modin: https://modin.readthedocs.io/en/latest/
- Swifter: https://pypi.org/project/swifter/
- Ray: https://pypi.org/project/ray/
- Pandarallel: https://pypi.org/project/pandarallel/
- Dask: https://dask.org/

- vectorization
- .iterrows()
- .apply()
- .itertuples()
- https://medium.com/swlh/why-pandas-itertuples-is-faster-than-iterrows-and-how-to-make-it-even-faster-bc50c0edd30d

Methodology	Average single run time	Marginal perform improvement
Crude looping	645 ms	
Looping with iterrows()	166 ms	3.9x
Looping with apply()	90.6 ms	1.8x
Vectorization with Pandas series	1.62 ms	55.9x
Vectorization with NumPy arrays	0.37 ms	4.4x

https://engineering.upside.com/a-beginners-guide-to-optimizing-pandas-code-for-speed-c09ef2c6a4d6