



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

The Dask documentation provides comprehensive guidance on parallel and distributed machine learning. The screenshot shows the "Dask Tutorial" page with a sidebar containing links to various topics like "Parallel and Distributed Machine Learning" and "Types of Scaling". The main content area features a section on "Parallel and Distributed Machine Learning" with a diagram illustrating a search space for hyperparameters.

You can run this notebook in a [live session](#) [launch binder](#) or view it [on Github](#).

Parallel and Distributed Machine Learning

[Dask-ML](#) has resources for parallel and distributed machine learning.

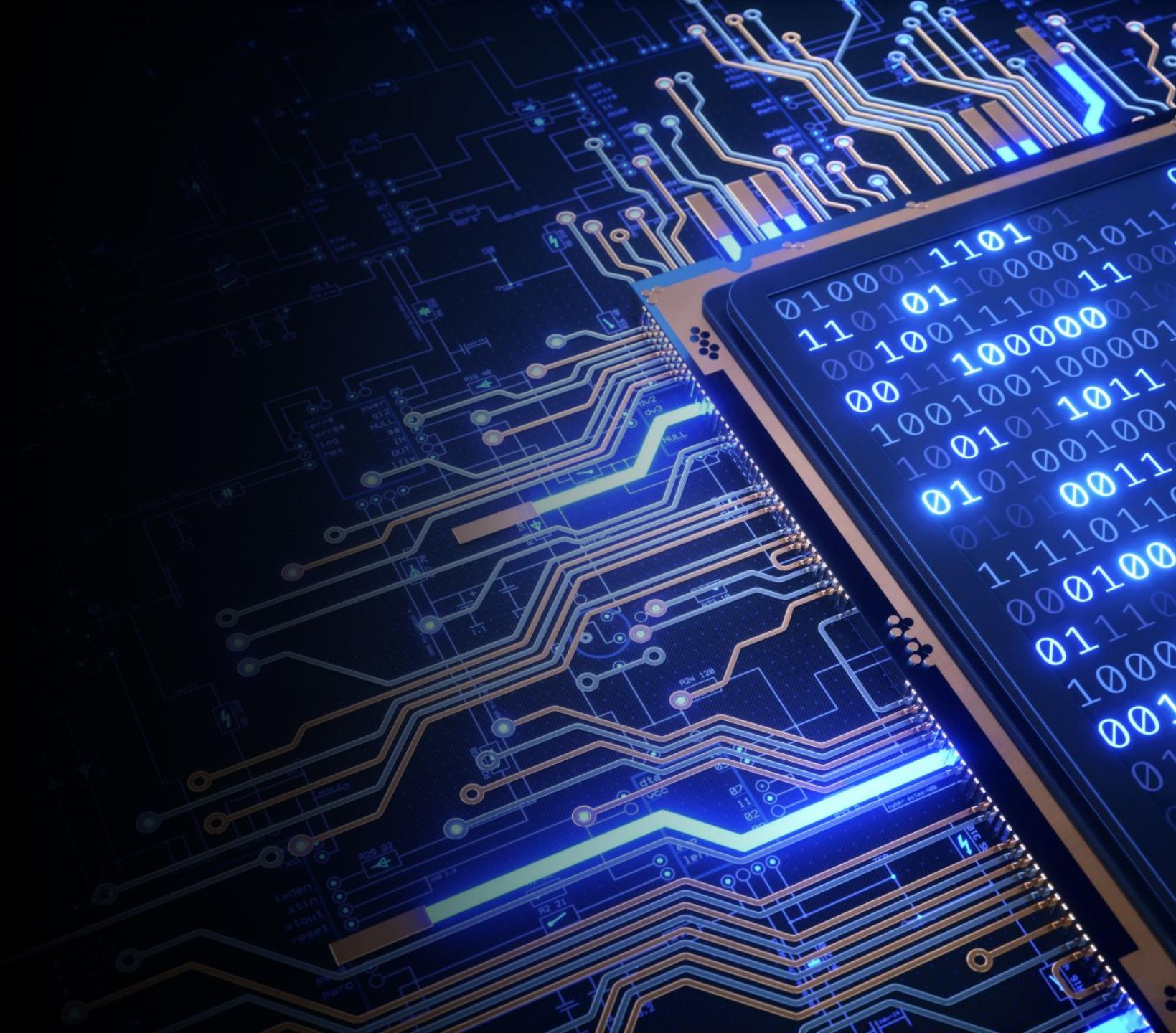
Types of Scaling

There are a couple of distinct scaling problems you might face. The scaling strategy depends on which problem you're facing.

1. CPU-Bound: Data fits in RAM, but training takes too long. Many hyperparameter combinations, a large ensemble of many models, etc.
2. Memory-bound: Data is larger than RAM, and sampling isn't an option.

```
graph TD; CBP[Choose Best Parameters] --> S1[SGDClassifier - alpha=1e-3]; CBP --> S2[SGDClassifier - alpha=1e-4]; CBP --> S3[SGDClassifier - alpha=1e-5]; CBP --> S4[SGDClassifier - alpha=1e-3]; CBP --> S5[SGDClassifier - alpha=1e-4]; CBP --> S6[SGDClassifier - alpha=1e-5]; T1[TfidfTransformer - norm='l1'] --> S1; T1 --> S3; T2[TfidfTransformer - norm='l2'] --> S4; T2 --> S5; T1 --> S6;
```

-
- **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.



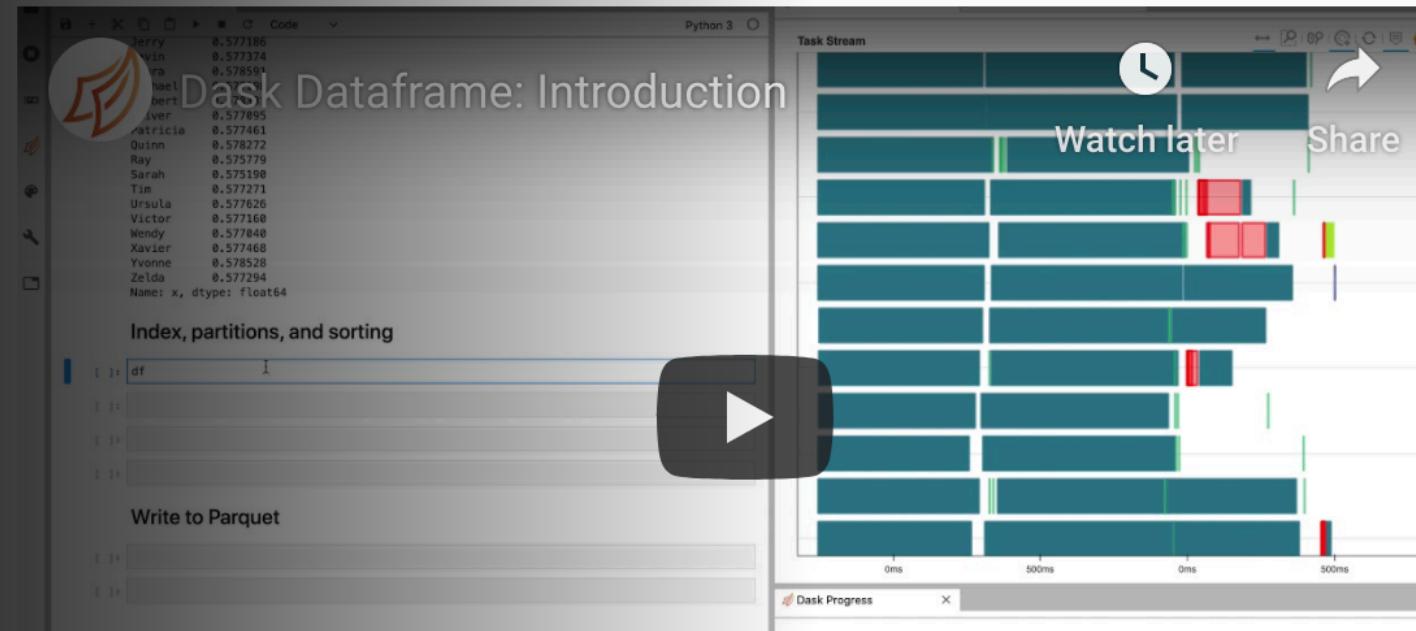


- Today's code was run on:
- Python virtualenv created with venv
- Requirements are in requirements.txt
- Python 3.7.6
- MacOS

DASK

DataFrame

A Dask DataFrame is a large parallel DataFrame composed of many smaller Pandas DataFrames split along the index. These Pandas DataFrames may live on disk for larger-than-memory data, on a single machine, or on many different machines in a cluster. One Dask DataFrame operation 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)

-
- 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>

