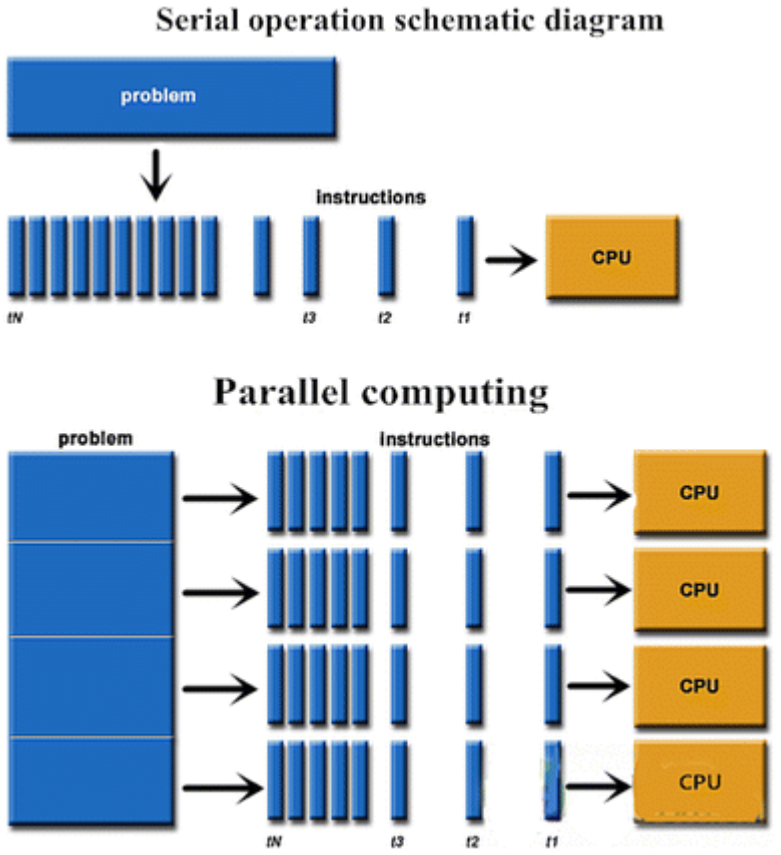
Student: Milrud Svetlana

Outline

Introduction

What is Dask

Dask is a flexible library for parallel computing in Python. It supports the Pandas dataframe and NumPy array data structures. It is able to either be run on your local computer or be scaled up to run on a cluster.



Need for Dask

Struggle with Pandas and NumPy while working with data which do not fit into RAM

https://docs.dask.org/en/latest/

Dask Opportunities

- Familiar user interface:

Dask DataFrame mimics Pandas

```
import pandas as pd                          import dask.dataframe as dd
df = pd.read_csv('2015-01-01.csv')           df = dd.read_csv('2015-*-*.csv')
df.groupby(df.user_id).value.mean()          df.groupby(df.user_id).value.mean().compute()
```

Dask Array mimics NumPy

```
import numpy as np                           import dask.array as da
f = h5py.File('myfile.hdf5')                 f = h5py.File('myfile.hdf5')
x = np.array(f['/small-data'])               x = da.from_array(f['/big-data'],
                                                               chunks=(1000, 1000))
x - x.mean(axis=1)                           x - x.mean(axis=1).compute()
```

Dask Bag mimics iterators, Toolz, and PySpark

```
import dask.bag as db
b = db.read_text('2015-*-*.json.gz').map(json.loads)
b.pluck('name').frequencies().topk(10, lambda pair: pair[1]).compute()
```

https://docs.dask.org/en/latest/

Dask Opportunities

- Scales from laptops to clusters:

  Dask is convenient on a laptop
  Dask can scale to a cluster of 100s of machines
  Easy transition between single-machine to moderate cluster

- Complex Algorithms:

  Dask represents parallel computations with task graphs

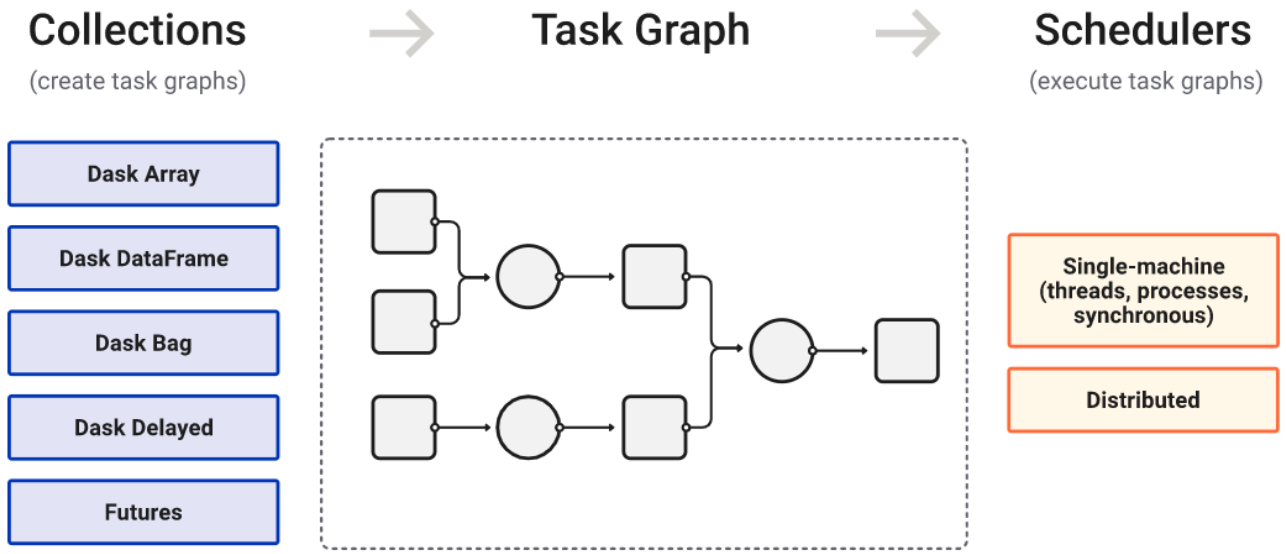
- Dask ML:

  Dask provides scalable machine learning in Python using Dask alongside popular machine learning libraries like Scikit-Learn, XGBoost, and others

https://docs.dask.org/en/latest/

Dask Collections

- High-level :
  - Dask Array
  - Dask Bag
  - Dask DataFrame
  - Dask ML
  - Others from external projects
- Low-level:
  - Delayed
  - Futures



High level collections are used to generate task graphs which can be executed by schedulers on a single machine or a cluster.

https://github.com/dask/dask-tutorial

Installation

DASK can be installed with conda, with pip or from source

conda

```
conda install dask  #This installs Dask and all common dependencies
```

```
conda install dask-core  #This installs Dask and minimum set of dependencies
```

pip

```
python  -m pip install “dask[complete]”  #This installs Dask and all common dependencies
```

```
python  -m pip install dask  #This installs Dask and minimum set of dependencies
```

source

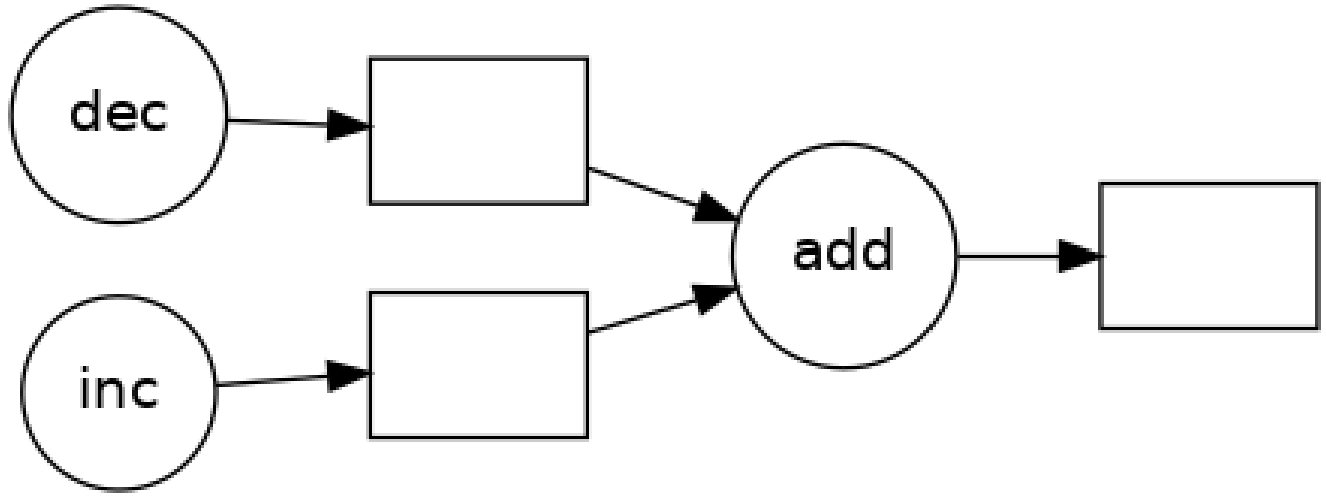
```
git clone https://github.com/dask/dask.git
cd dask python
-m pip install .
```

https://docs.dask.org/en/stable/install.html

Dask Delayed

Sometimes problems do not fit into one of the collections like `dask.array` or `dask.dataframe`. In these cases, users can parallelize custom algorithms using the simpler `dask.delayed` interface

Follow code in the notebook



https://examples.dask.org/delayed.html

Dask Future

Dask futures provide fine-grained real-time execution for custom situations. This is the foundation for other APIs like Dask arrays and dataframes

Unlike for `arrays` and `dataframes`, you need the Dask client to use the Futures interface

```
[1]: from dask.distributed import Client, progress
     client = Client(threads_per_worker=4, n_workers=1)
     client
```

Follow code in the notebook

https://examples.dask.org/futures.html

Dask Bag

Dask Bag implements operations like `map`, `filter`, `groupby` and aggregations on collections of Python objects. It does this in parallel and in small memory using Python iterators. It is similar to a parallel version of itertools or a Pythonic version of the PySpark RDD

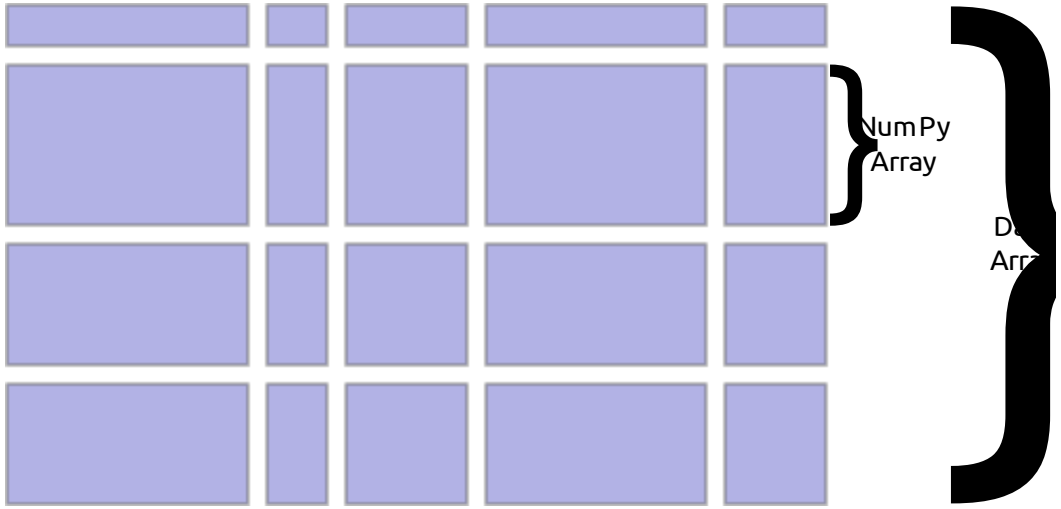
Follow code in the notebook

https://examples.dask.org/bag.html

Dask Array

Dask array provides a parallel, larger-than-memory, n-dimensional array using blocked algorithms.

- **Parallel**: Uses all of the cores on your computer
- **Larger-than-memory**: Lets you work on datasets that are larger than your available memory by breaking up your array into many small pieces, operating on those pieces in an order that minimizes the memory footprint of your computation, and effectively streaming data from disk.
- **Blocked Algorithms**: Perform large computations by performing many smaller computations



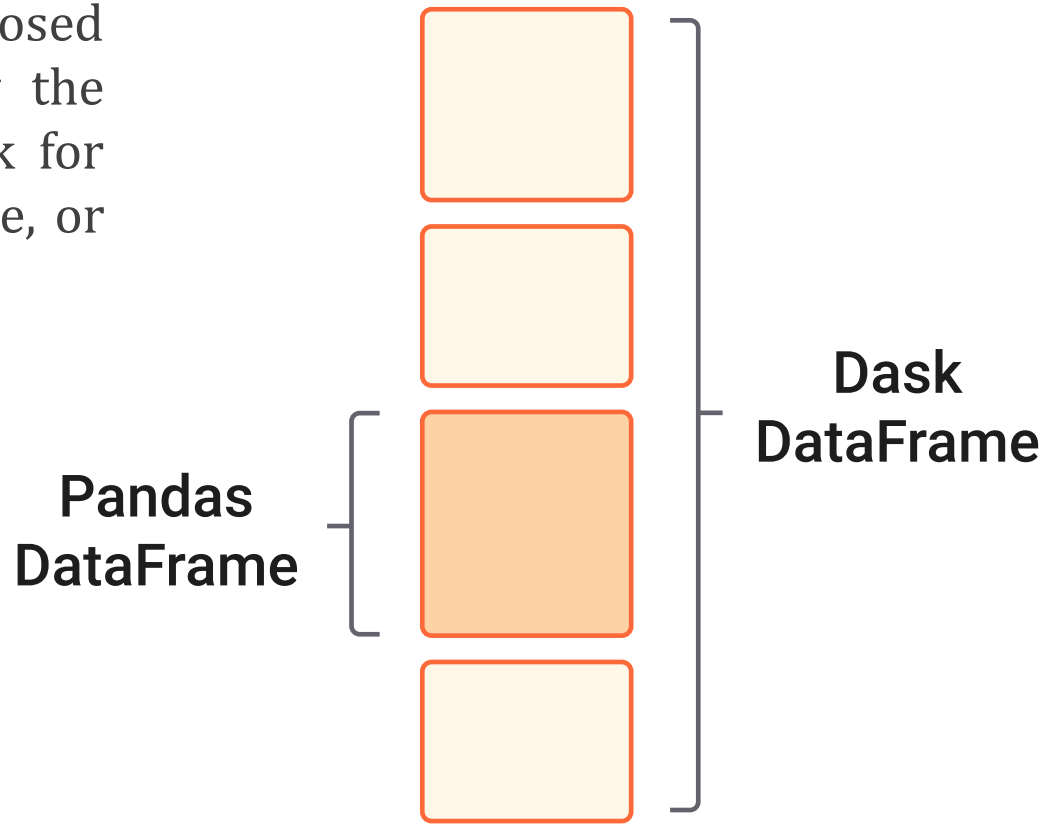
Follow code in the notebook

https://examples.dask.org/array.html

Dask DataFrame

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 computing on a single machine, or on many different machines in a cluster

Follow code in the notebook



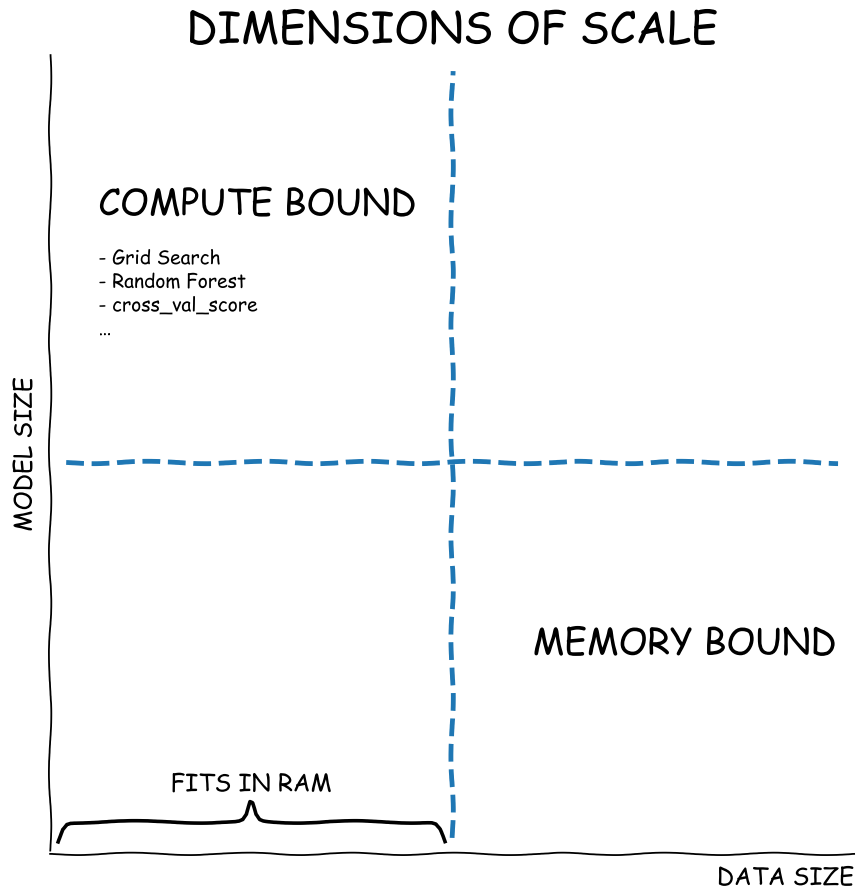
https://docs.dask.org/en/stable/dataframe.html

Dask ML

Dask-ML provides scalable machine learning in Python using Dask alongside popular machine learning libraries like Scikit-Learn, XGBoost, and others.

Challenge 1: Scaling Model Size
When models growing so large or complex that tasks like model training, prediction, or evaluation steps will take too long to complete

Challenge 2: Scaling Data Size
When datasets grow larger than RAM so loading the data into NumPy or pandas becomes impossible



https://ml.dask.org/

Dask ML

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

- For in-memory problems, just use scikit-learn (or your favorite ML library)
- For large models, use dask_ml.joblib and your favorite scikit-learn estimator
- For large datasets, use dask_ml estimators

See more detailed information in in the notebook

https://ml.dask.org/

Dask ML - Training on Large Datasets

- Most estimators in scikit-learn are designed to work on in-memory arrays. Training with larger datasets may require different algorithms
- All of the algorithms implemented in Dask-ML work well on larger than memory datasets, which you might store in a dask array or dataframe

[9]:
```
%matplotlib inline
```

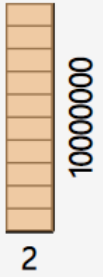
[10]:
```
import dask_ml.datasets
import dask_ml.cluster
import matplotlib.pyplot as plt
```

[11]:
```
X, y = dask_ml.datasets.make_blobs(n_samples=10000000,
                                   chunks=1000000,
                                   random_state=0,
                                   centers=3)

X = X.persist()
X
```

[11]:

	Array	Chunk
Bytes	152.59 MiB	15.26 MiB
Shape	(10000000, 2)	(1000000, 2)
Count	10 Tasks	10 Chunks
Type	float64	numpy.ndarray



In this example, dask_ml.datasets.make_blobs generates some random dask arrays

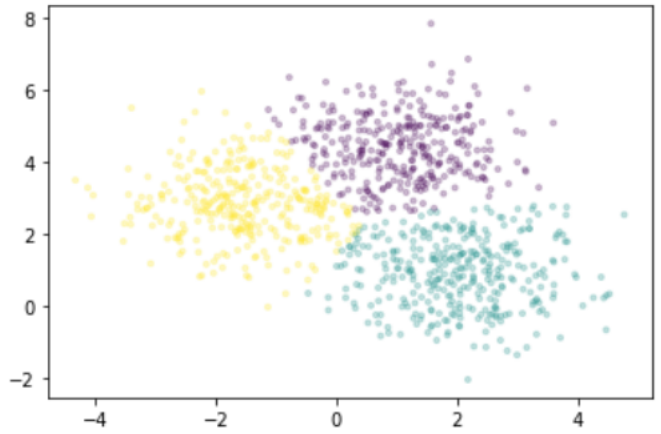
Dask ML - Training on Large Datasets


```
[12]: km = dask_ml.cluster.KMeans(n_clusters=3, init_max_iter=2, oversampling_factor=10)
      km.fit(X)
```

```
/usr/share/miniconda3/envs/dask-examples/lib/python3.9/site-packages/dask/base.py:1283: UserWarning:
  warnings.warn(
```

```
[12]: KMeans(init_max_iter=2, n_clusters=3, oversampling_factor=10)
```

```
[13]: fig, ax = plt.subplots()
      ax.scatter(X[::10000, 0], X[::10000, 1], marker='.', c=km.labels_[::10000],
                 cmap='viridis', alpha=0.25);
```



k-means implemented in Dask-ML is used to cluster the points.

Plot a sample of points, colored by the cluster each falls into

https://ml.dask.org/

Dask ML- Generalized Linear Models

Generalized linear models are a broad class of commonly used models. These implementations scale well out to large datasets either on a single machine or distributed cluster

LinearRegression([penalty, dual, tol, C, ...])	Estimator for linear regression.
LogisticRegression([penalty, dual, tol, C, ...])	Estimator for linear regression.
PoissonRegression([penalty, dual, tol, C, ...])	Estimator for linear regression.

Example

```
In [1]: from dask_ml.linear_model import LogisticRegression

In [2]: from dask_ml.datasets import make_classification

In [3]: X, y = make_classification(chunks=50)

In [4]: lr = LogisticRegression()

In [5]: lr.fit(X, y)
Out[5]: LogisticRegression()
```

https://ml.dask.org/

Dask ML

Pipelines and Composite Estimators

Dask-ML estimators follow the scikit-learn API. This means Dask-ML estimators like dask_ml.decomposition.PCA can be placed inside a regular sklearn.pipeline.Pipeline

Example

```
In [1]: from sklearn.pipeline import Pipeline  # regular scikit-learn pipeline

In [2]: from dask_ml.cluster import KMeans

In [3]: from dask_ml.decomposition import PCA

In [4]: estimators = [('reduce_dim', PCA()), ('cluster', KMeans())]

In [5]: pipe = Pipeline(estimators)

In [6]: pipe
Out[6]: Pipeline(steps=[('reduce_dim', PCA()), ('cluster', KMeans())])
```

http://scikit-learn.org/dev/modules/compose.html

Dask ML - Scikit-Learn & Joblib

Many Scikit-Learn algorithms are written for parallel execution using Joblib which natively provides thread-based and process-based parallelism

Dask can scale these Joblib-backed algorithms out to a cluster of machines by providing an alternative Joblib backend

See example in in the notebook

https://ml.dask.org/joblib.html

Dask ML - XGBoost & LightGBM

Dask-ML can set up distributed XGBoost or LightGBM for you and hand off data from distributed dask.dataframes. This automates much of preprocessing and setup while still letting XGBoost/LightGBM do what they do well

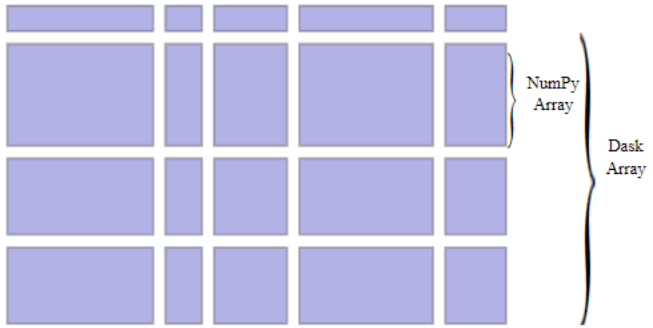
train(client, params, data, labels[, ...])	Train an XGBoost model on a Dask Cluster
predict(client, model, data)	Distributed prediction with XGBoost
XGBClassifier(*[, objective, use_label_encoder])	Attributes
XGBRegressor(*[, objective])	Attributes

https://ml.dask.org/xgboost.html

Conclusion

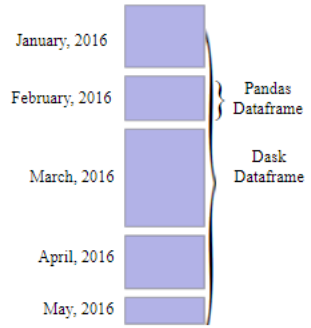
The advantage of using Dask is that **it can scale computations to multiple cores on your computer**. This enables work on large datasets that don't fit into memory. It also aids in speeding up computations that would ordinarily take a long time. Dask also provides ability for machine learning for training and running predictions

NumPy



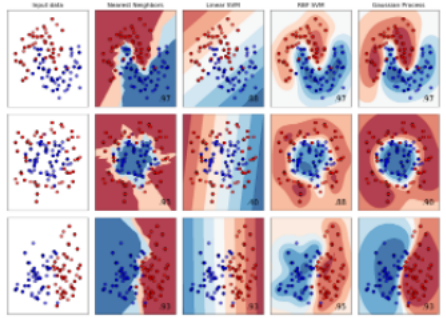
Dask arrays scale NumPy workflows, enabling multi-dimensional data analysis in earth science, satellite imagery, genomics, biomedical applications, and

pandas



Dask dataframes scale pandas workflows, enabling applications in time series, business intelligence, and general data munging on big data.

scikit-learn



Dask-ML scales machine learning APIs like scikit-learn and XGBoost to enable scalable training and prediction on large models and large datasets.

Thank you for attention!