Parallel Machine Learning and Artificial Intelligence

Dr. Handan Liu

h.liu@northeastern.edu

Northeastern University

Northeastern University
College of Engineering

Multiprocessing Pool Examples

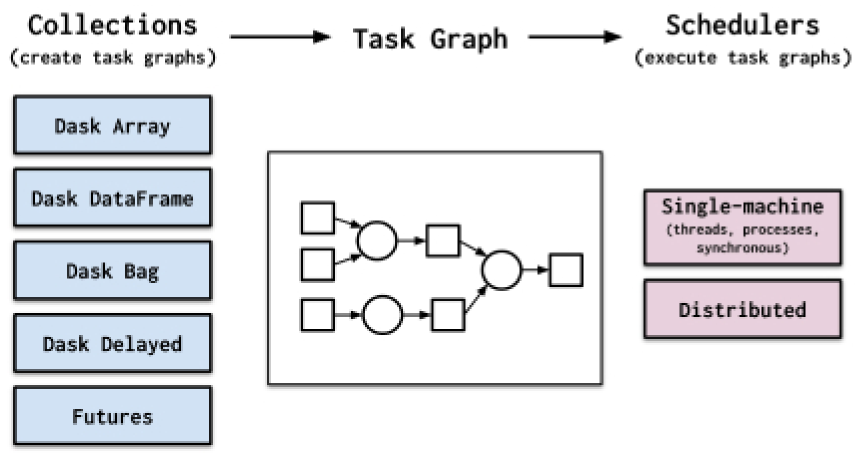
- Example: given a serial code
  - Parallelized in Pool.map()
  - Parallelized in Pool.apply()
  - Parallelized in Pool.starmap()

- The above methods in Pool class take the function to be parallelized as the main argument.
- The map takes only one iterable as an argument.
- In starmap(), each element in that iterable is also an iterable.
- The apply call function with arguments *args* and keyword arguments *kwds*.

Why Dask?

- Python's role in Data Science: Python has grown to become the dominant language both in data analytics and general programming, but limited to use on a single CPU on a single machine.
- Dask is open source and freely available. It is developed in coordination with widely used tools like NumPy, pandas, and scikit-learn, to scale them to multi-core machines and distributed clusters.
- Dask scales out to clusters
- Dask scales down to single machines

Execution



By default, Dask Array uses the threaded scheduler in order to avoid data transfer costs, and because NumPy releases the GIL well. It is also quite effective on a cluster using the dask.distributed scheduler.

Install Dask

- You can install dask with conda, with pip, or by installing from source.

- Dask is installed by default in Anaconda. You can update Dask using the conda command:
  - conda install dask

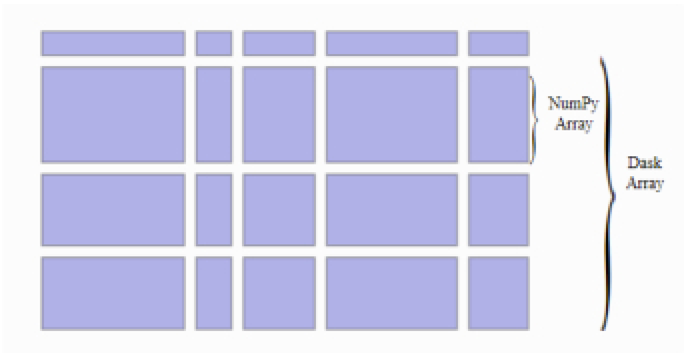
- This installs Dask and all common dependencies, including Pandas and NumPy.

Install Dask-labextension

- This package provides a JupyterLab extension to manage Dask clusters, as well as embed Dask's dashboard plots directly into JupyterLab panels.
- Install dask-labextension:
  - Jupyterlab 3.0 + : `conda install -c conda-forge dask-labextension` or `pip install dask-labextension`
  - Jupyterlab 2.x:
    - ✔ The installation is more complicated. You must install Node.js version >= 12, and separately install client-side and server-side components of this package.
    - ✔ Suggest to upgrade to Jupyterlab3
      - install and use extensions faster and more conveniently
      - It does not require users to rebuild JupyterLab or install Node.js
      - Install extension through pip
- Install jupyterlab3:
  - Command: `conda install -c conda-forge jupyterlab=3` or `pip install jupyterlab==3`

Dask Array mimics NumPy

Dask Array implements a subset of the NumPy ndarray interface using *blocked algorithms*, cutting up the large array into many small arrays.



- Dask arrays coordinate many Numpy arrays, arranged into chunks within a grid. They support a large subset of the Numpy API.
- We coordinate these *blocked algorithms* using Dask graphs.

```python
import numpy as np
f = h5py.File('myfile.hdf5')
x = np.array(f['/small-data'])

x - x.mean(axis=1)
```

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

- Dask arrays are composed of many NumPy (or NumPy-like) arrays. How these arrays are arranged can significantly affect performance.
- Different arrangements of NumPy arrays will be faster or slower for different algorithms.
- Thinking about and controlling chunking is important to optimize advanced algorithms.

Specifying Chunk shapes

1. A uniform dimension size like 1000 for each dimension;
2. A uniform chunk shape like (1000, 2000, 3000) for 1st, 2nd, 3rd axis respectively;
3. Fully explicit sizes of all blocks along all dimensions, like ((1000, 1000, 500), (400, 400), (5, 5, 5, 5, 5))
4. A dictionary specifying chunk size per dimension like {0: 1000, 1: 2000, 2: 3000}. This is just another way of writing the forms 2 and 3 above

Rules for selecting the size of “chunks”

1. A chunk should be small enough to fit comfortably in memory. We’ll have many chunks in memory at once.

2. A chunk must be large enough so that computations on that chunk take significantly longer than the 1ms overhead per task that Dask scheduling incurs. A task should take longer than 100ms.

3. Chunk sizes between 10MB-1GB are common, depending on the availability of RAM and the duration of computations.

4. Chunks should align with the computation that you want to do.

5. Chunks should align with your storage, if applicable.

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 computing on a single machine, or on many different machines in a cluster.
- One Dask DataFrame operation triggers many operations on the constituent Pandas DataFrames.

Example: Dask DataFrame

Execution

- By default, Dask DataFrame uses the multi-threaded scheduler. This exposes some parallelism when Pandas or the underlying NumPy operations release the global interpreter lock (GIL).
- Generally, Pandas is more GIL bound than NumPy, so multi-core speedup on multi-core is not as pronounced for Dask DataFrame as they are for Dask Array.
- This is changing, and the Pandas development team is actively working on releasing the GIL.

- Stay safe!
- See you next class!

<u>Next Lecture will Continue:</u>

Parallel Machine Learning