HPS

Zoya Masih
Supervisor: Prof. J. Kunkel

Analyzing I/O performance when using DASK for ML

Out-of-Core ML

Table of contents

Table of Contents

What is Out-of-Core Machine Learning

- Out-of-Core algorithms process too large data sets
  - i.e., the data that can’t fit into the main memory
- Analyzing social media data is an example
  - To detect trends, and understand customer preferences
- Training a big model like ChatGPT is another example

Motivation

- When a cluster runs out of memory, it can lead to various issues and errors:
  - Slowdowns
  - System instability
  - Crashes, or failure to allocate memory for new processes or tasks.
- This impacts the performance, reliability, and job executions of the cluster
- Out-of-core ML:
  - leverages disk-based storage and streaming algorithms,
  - enables scalable and parallel processing

How Out-of-Core ML works

- The ability to learn incrementally (online learning) is key to out-of-core ML
- In traditional ML, the training model has access to the whole data set
- The model learns from all training data at once
  - The process may iterate for several epochs

How Out-of-Core ML works

- In incremental learning, models learn from new info in real-time or on-the-fly
- The model starts learning with a small subset which fits into RAM
- When new data becomes available, the model is updated
  - while preserving its existing knowledge

Table of Contents

What is Dask

- A library for parallel computing in Python.
- It can be used on a workstation or a huge cluster

What is Dask

- Parallelize any Python code, letting you scale any function
- Makes it easy to scale the Python libraries

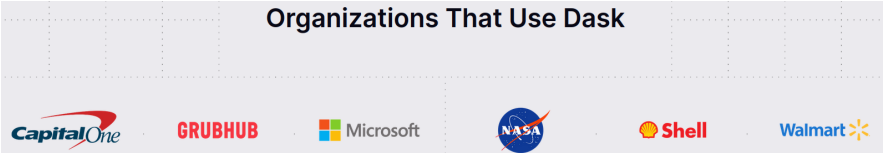


Figure: https://www.dask.org/

Dask General Code Example

- Numpy and Dask arrays

```
1 import numpy as np
2 x = np.ones(15)
3 x
```

»array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

Dask General Code Example

- Numpy and Dask arrays

```
1 import numpy as np
2 x = np.ones(15)
3 x
```

»array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])

```
1 import dask as da
2 x = da.array.ones(15,chunks=(5,))
3 x
```

	Array	Chunk
Bytes	120 B	40 B
Shape	(15,)	(5,)
Count	3 Tasks	3 Chunks
Type	float64	numpy.ndarray

Dask for ML

- Dask also provides strong tools for out-of-core ML
- Dask-ML provides scalable machine learning in Python using Dask
  - Beside ML libraries like Scikit-Learn, XGBoost, etc.

```
1 from dask_ml.xgboost import XGBRegressor
2
3 est = XGBRegressor(...)
4 est.fit(train, train_labels)
```

Dask-ML Code Example

- Algorithms implemented in Dask-ML work well on larger than memory datasets, stored in a dask array or dataframe.

```
1 import dask_ml.datasets
2 import dask_ml.cluster
3 import matplotlib.pyplot as plt
4 X, y = dask_ml.datasets.make_blobs(n_samples=10000000, chunks=1000000,
      ↪ random_state=0, centers=3)
5 X = X.persist()
6 X_np = X.compute()
7 y_np = y.compute()
```

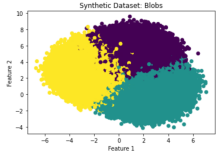


Table of Contents

Project plans

- Developing a model to predict the species of trees in a forest

Project plans

- Developing a model to predict the species of trees in a forest
- Regarding the IO-intensive nature of the job, I run the ML pipeline on SCC.
  - To evaluate the IO performance

Project plans

- Developing a model to predict the species of trees in a forest
- Regarding the IO-intensive nature of the job, I run the ML pipeline on SCC.
  - To evaluate the IO performance
- Additional jobs that are desirable to be accomplished:
  - Develop a more robust and accurate model
  - Testing different configurations to identify how to optimize I/O there

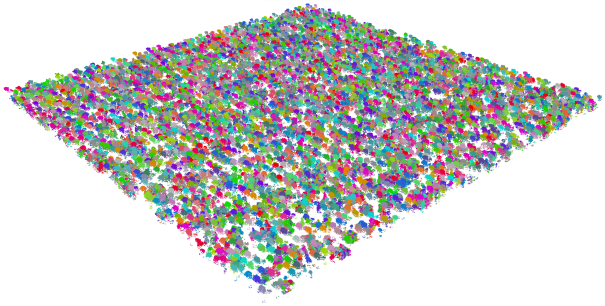
The first step: Data Preprocessing

- The forest dataset has been sourced from a GWDG GitLab repository
  - Belonging to Ali Doost Hosseini
  - https://gitlab-ce.gwdg.de/adoosth/synforest/-/tree/main

The first step: Data Preprocessing

- The forest dataset has been sourced from a GWDG GitLab repository
  - Belonging to Ali Doost Hosseini
  - https://gitlab-ce.gwdg.de/adoosth/synforest/-/tree/main
- SynForest is a tool that generates realistic large-scale point clouds of forests
  - By simulating the Lidar scanning process
  - from a stationary or moving platform
  - By Capturing the shape and structure of realistic tree models.
- Lidar: Light Detection and Ranging
  - Uses light (pulsed laser) to measure variable distances to the Earth

The laz file



Preprocessing

- We needed to fit the created forests to the ML model requirements
  - The forest was including just one type of tree
  - The created laz file doesn't include the tree's data
    - We added a CSV file, including the tree attributes

Loading data

```
1 import dask.dataframe as dd
2 import laspy
3
4 las = laspy.read('/home/zoya/forest1.laz')
5 df = dd.read_csv('/home/zoya/forest1.csv')
6
7 print(len(las.points))
8 #output: 5544915
```

- Dask dataframes

```
X_las = dd.from_pandas(pd.DataFrame({'X': las.x, 'Y': las.y, 'Z': las.z,
    ↪ 'hitObjectId': las.hitObjectId}), npartitions=1)
X_csv = df.loc[0:533, ['d', 'h', 'spec','hitObjetId']].values
a = X_csv[0, 0].compute()
b = X_csv[0, 1].compute()
c = X_csv[0, 2].compute()
d = X_csv[0, 3].compute()
X_csv_dd = dd.from_dask_array(X_csv, columns=['d','h','s','hitObjectId'])
```

first tree, diameter: 0.285
first tree, height: 24.50475
first tree, species: PicAbi
first tree, ID: 0

Merging the files

```
import dask.dataframe as dd

x = dd.from_pandas(pd.DataFrame({'X': ['x11', 'x12', 'x13', 'x14','x15'],
    ↪ 'Y': ['y12', 'y22', 'y23','y24', 'y25'],'Z': ['z13', 'z23',
    ↪ 'z33','z34','z35'], 'id': [1, 1, 0,2,0]}), npartitions=1)
y = dd.from_pandas(pd.DataFrame({'d': ['d1', 'd2', 'd3'], 'h': ['h1', 'h2',
    ↪ 'h3'],'s': ['s1','s2','s3'], 'id': [0,1,2]}), npartitions=1)
result = dd.merge(x, y, on='id')
result = result.drop('id', axis=1)
result.compute()
```

The laz file:

	X	Y	Z	id
0	x11	y12	z13	1
1	x12	y22	z23	1
2	x13	y23	z33	0
3	x14	y24	z34	2
4	x15	y25	z35	0

The CSV file:

	d	h	s	id
0	d1	h1	s1	0
1	d2	h2	s2	1
2	d3	h3	s3	2

The laz file:

	X	Y	Z	id
0	x11	y12	z13	1
1	x12	y22	z23	1
2	x13	y23	z33	0
3	x14	y24	z34	2
4	x15	y25	z35	0

The CSV file:

	d	h	s	id
0	d1	h1	s1	0
1	d2	h2	s2	1
2	d3	h3	s3	2

	X	Y	Z	d	h	s
0	x11	y12	z13	d2	h2	s2
1	x12	y22	z23	d2	h2	s2
2	x13	y23	z33	d1	h1	s1
3	x15	y25	z35	d1	h1	s1
4	x14	y24	z34	d3	h3	s3

References

- https://www.analyticsvidhya.com/blog/2022/09/out-of-core-ml-an-efficient-technique-to-handle-large-data/
- https://subscription.packtpub.com/book/web-development/9781785887215/2/ch02lvl1sec12/out-of-core-learning
- https://medium.com/productive-data-science/out-of-core-larger-than-ram-machine-learning-with-dask-9d2e5f29d733
- http://ml.dask.org/index.html https://medium.com/when-i-work-data/processing-data-with-dask-47e4233cf165 https://scikit-learn.org/0.15/modules/scaling_strategies.html