Parallel Programming with Dask:

Course Proposal to Datacamp

Swastik Nath.

CHAPTER 1: DASK ARRAYS:

- Lesson 1.1: An Introduction to Dask Arrays

  Learning Objective:
  - Learner will be able to scope of using Dask Arrays, will be able to understand how it is implemented internally, understand its related attributes like chunks, blocks and how dask manages to perform computation on larger than memory arrays.

- Lesson 1.2: Creating and Storing Dask Arrays

  Learning Objective:
  - Learners will be able to understand and implement Dask Arrays from different supported collections and sources including other dask collections, hdf5, s3 object bucket, parquet, avro etc. and also be able to store dask arrays into different supported collections and formats like .npy files, zarr, hdf5 etc.
  - Learners will be able to read a large number of images to dask arrays.
  - Learners will get a comprehensive overview of Dask Array Chunks, at the end of this chapter they will be able to use clever chunking or the automatic chunking to be able to implement efficient algorithm.

  Some functions used: from_array(), from_npy_stack(), from_delayed(), to_hdf5(), to_npy_stack(), compute_chunk_sizes(), image.imread()

- Lesson 1.3: Performing Familiar Numpy functions over Dask Arrays

  Learning Objective:
  - Learners will be able to apply several familiar numpy functions at scale over the dask arrays.
  - Learners will be able to concatenate, broadcast, stack, split multiple dask arrays.
  - Learners will be able to use argtopk, topk to efficiently obtain sorted dask arrays.

Some functions used: broadcast_to(), broadcast_arrays(), concatenate(), argtopk(), topk(), gradient(), hstack(), stack(), vstack(), split() etc.

- Lesson 1.4: Performing Statistical Analysis and Linear Algebra at scale.
  Learning Objectives:
  - Learners will be able to use dask.array.stats perform several statistical analyses for example kurtosis, skew, moment.
  - Learners will be able to use the dask.array.random module to perform sampling from statistical distributions.
  - Learners will be able to use the dask.array.linalg to perform certain specific parallel friendly implementation of Linear Algebra for example LU Decompositions, QR factorization etc. at larger than memory datasets.

  Some functions used: linalg.norm(), linalg.qr(), linalg.lu(), stats.kurtosis(), stats.skew(), etc.

CHAPTER 2: DASK BAGS:

- Lesson 2.1: An Introduction to Dask Bags

  Learning Objectives:
  - Learners will be able to understand the significance of Bags as collection, its design approach, use-cases and limitations in certain parallelization scenario etc.
  - Learners will also be able to comprehend the concept of partitions and its relevance in handling larger than memory Bags.

- Lesson 2.2: Creating and Storing Dask Bags

  Learning Objectives:
  - Learners will be able to understand and implement Dask Bags from different supported collections and sources including other dask collections like Dask Dataframes, python sequence, text files, urls, avro etc. and also be able to store dask bags into different supported collections and formats like dataframes, delayed objects, test files.

  Some functions used: from_sequence(), read_text, read_avro(), from_delayed(), to_textfiles(), to_avro(), to_dataframe() etc.

- Lesson 2.3: Performing Operations over Bags in Parallel

  Learning Objectives:
  - Learners will be able to understand and implement Bag operations like repartitioning counting distinct elements, frequencies, flatten, plucking, filtering, basic statistical methods like standard deviations, mean, max etc.
  - Learners will be able to join multiple Bag Collections, and removing specific elements from Bags efficiently.
  - Learners will also be able to randomly sample bags using dask.bag.random
  - Learners will be able to visualize the Dask Task graphs with visualize()

  Some functions used: merge(), visualize(), frequency(), count(), distinct(), pluck(), take(), repartition(), remove()

- Lesson 2.4: Performing Parallelized Aggregations and Reductions

  Learning Objectives:
  - Learners will be able to apply custom functions elementwise or partition wise.
  - Lerarners will be able to perform parallel reductions and aggregations with fold and foldby() with details why these two are preferrable in a scaled out situation instead of groupby()

  Some functions used: foldby(), fold(), map(), map_partitions()

Chapter 3: Dask Dataframes:

- Lesson 3.1: An Introduction to Dask Dataframes

  Learning Objectives:
  - Learners will be able to comprehend how Dask manages to handle massive out of memory datasets efficiently with a small discussion around its clever implementation, use cases, benefits for parallel perform parallel tasks.
  - Learners will be able to understand and implement Dask Dataframes from different supported collections and sources including other dask collections like Dask Arrays, Bags, Pandas Dataframes, csv files, hdf5, parquet, json, sql tables etc. and also be able to store

dask bags into different supported dask collections and formats like parquet, csv, sql, json etc.
- Learners will be able to modify dataframe properties like number of partitions, slice along partitions, view memory usages.

Some functions used: read_csv(), read_parquet(), read_sql_table(), read_orc(), read_json() etc.

- Lesson 3.2: Performing Familiar Pandas Operations over Dask Dataframes at Scale

Learning Objectives:
- Learners will be able to perform parallel implementation of Pandas functions like head, describe, drop, fill_na, set_index and more over a massive larger than memory dataset.
- Learners will be able to perform several statistical aggregations like standard deviation, mean, mode, max, min etc. Learners will also be able to perform cumulative aggregations for example cumsum, cummax, cummin etc. over any specified axis.

Some functions used: head(), describe(), tail(), fill_na(), cumsum(), cummax(), mean(), mode() etc.

- Lesson 3.3: Performing Parallelization-friendly aggregations and reductions at scale.

Learning Objectives:
- Learners will be able to perform common aggregations with groupby at scale in parallel.
- Learners will be able to perform computationally expensive tasks like drop_duplicates, value_counts on a massive scale.
- Learners will be able to merge two dask dataframes or with a Pandas dataframe leveraging the distributed platform.
- Learners will be able to perform date-time resampling over a large dataset.
- Learners will be able to perform rolling window aggregations at a massive scale in parallel.

Some functions used: groupby(), apply(), drop_duplicates(), value_counts(), merge(), resample(), rolling(), corr(), set_index(), isin(), etc.

CHAPTER 4: DASK DISTRIBUTED SCHEDULERS, DASK DELAYED, AND FUTURES:

- Lesson 4.1: Working with Dask's Distributed Schedulers

  Learning Objectives:

  - Learners will be able to leverage dask's distributed schedulers to parallelize their workloads across a cluster of machines or within a single machine (LocalCluster()) but across all the CPU cores.
  - Learners will be able to visualize their tasks execution on a diagnostics dashboard called the Dask Dashboard. Learners will be able to generate task performance reports and task execution reports.

  Some functions used: performance_report(), Client(), LocalCluster(), get_task_stream()

- Lesson 4.2: Writing Custom parallelizable functions with Dask Delayed

  Learning Objectives:

  - Learners will be able to comprehend the concept of Dask Delayed in the context of Dask Task graphs and will be implement custom functions which can be scaled to run over a massive dataset with Dask Delayed. Learners will learn about two implementation method of Dask Delayed, as a wrapper and as a decorator.

  Some functions used: delayed()

- Lesson 4.3: Working with Futures

  Learning Objectives:

  - Learners will be able to get a first hand look at the concepts and implementations of Dask Futures which can be used to schedule a task in the background parallelly and will be able interact with the remotely running task.

  Some functions used: Future.result(), Client.gather(), Client.submit(), Future.traceback(), Future.cancel(),etc.

CHAPTER 5: SCALABLE MACHINE LEARNING WITH DASK

- Lession 5.1: Processing Massive Dataset for Machine Learning:

  Learning Objectives:

  - Learners will be able to split a massive dataset into training and test dataset in parallel.
  - Learners will be able to reduce the dimensionality of a large out-of-memory dataset using PCA.

- Learners will be able to normalize the features in a massive dataset.

Some functions used: train_test_split(), PCA(), MinMaxScalar()

- Lession 5.2: Scalable Regression and Classification with dask-ml Estimators:

Learning Objectives:

- Learners will be able to build scalable estimators which can perform training and inference parallelly.

Some functions used: XGBClassifier(), XGBRegressor(), LinearRegression(), LogisticRegression(), KMeans()

- Lession 5.3: Scaling out Scikit learn models with Parallel Meta Estimators

Learning Objectives:

- Learners will be able to implement ParallelPostFit and Incremental meta estimators which scales out non-parallel scikit learn models to parallelly perform predict(), predict_proba() and fit(), predict() and predict_proba() respectively, thereby parallelizing training and inferencing over a large dataset.

Some functions used: ParallelPostFit(), Incremental(), sklearn.svm.SVC(), sklearn.linear_model.SGDClassifier()

- Lession 5.4: Parallel Hyperparameter Search with dask-ml

Learning Objectives:

- Learners will be able to build hyperparmeter tuner which scales out for even out-of-memory dataset and uses the parallel meta estimators for parallelizing training and inferencing for out of memory datasets. Learners will be able to implement Dask's implementation of GridSearchCV, RandomizedSearchCV and IncrementalSearchCV().

Some functions used: GridSearchCV(), RandomizedSearchCV(), IncrementalSearchCV()