PARALLEL COMPUTATION WITH BLOCKED ALGORITHMS AND TASK SCHEDULING
Problem Statement

• Memory in our computers is limited
• Expensive to expand
• The size of datasets to analyze is growing exponentially
  ◦ Social Networking
  ◦ Online Shopping
• Not everyone has access to a cluster
• What do you do when you cannot load your data into memory?
What is Dask?

• A library for parallel computing
• Only supported in Python
• Scales from laptop to cluster
• Out of core algorithms
• Blocked algorithms
• The three focal points of Dask
  • Dask Collections
  • Task Graphs
  • Dynamic Task Scheduling
Dask Array

• A n-dimensional array
• A collection of NumPy arrays that uses blocked algorithms
• Mimics the NumPy API
• Does not support the entirety of the NumPy interface
• Dask array functions produce Array objects that hold on to task graphs.
• Implement the __array__ protocol we can use them directly into functions of other libraries

The number and size of the Numpy Array in the collection is customizable. These are called chunks.

The Dask Array is not the first library to implement a NumPy clone.

Biggus, an out-of-core ndarray specialized for climate science, Spartan, Theano

Each of these implementations focuses on a particular application or domain.

Dask Array distinguishes itself in that it is a general class of NumPy operations and execution through dynamic task scheduling.
Dask DataFrame

• A collection of Pandas DataFrames
• Mimics Pandas API
• Does not support the entirety of the DataFrame interface
• Some Pandas operations do not release the GIL
• In some cases, parallelism is not possible
• May or may not be the case in the future.
• Computes more efficiently on partitioned datasets where the different blocks are well separated along an index
Blocked Algorithms

• The objective of blocked algorithms to compute large operations efficiently
  • Example: Sum two 1,000,000x1,000,000 matrices

• Use many computations to sum small chunks, then take the sum of all those chunks

• Allows an opportunity for collections to use all of your cores

• Blocked algorithms have proven useful in modern numerical linear algebra libraries like Flame and Plasma and more recently in data parallel systems like Dryad and Spark

• Datasets may not be able to fit in memory

• **Out of core Algorithms** - Process data moving in and out of memory from external storage

• Allows collections to process data too big for memory

• In this case, NumPy arrays or Pandas DataFrames are being moved to and from memory to process to carry out computations on datasets efficiently which may or may not be too big for memory
Task Graphs

- Used to encode dynamic task scheduling
- Represent the structure of our program itself explicitly as data
- Represented by an internal python dictionary that maps keys to values and tasks
- Created by operations on Dask collections
- Keys are our hashable variables
- A task is a tuple with a callable first element.
- Tasks represent atomic units of work meant to be run by a single worker.
- Succeeding elements are arguments for that function.
- An argument may be one of the following:
  1. Any key
  2. Any other value like 1, to be interpreted literally
  3. Other tasks
  4. List of arguments

Consider the following simple program:

```python
def inc(i):
    return i + 1

def add(a, b):
    return a + b

x = 1
y = inc(x)
z = add(y, 10)
```

We encode this as a dictionary below:

```python
d = {
    'x': 1,
    'y': (inc, 'x'),
    'z': (add, 'y', 10)
}
```
Dask then executes this graph in parallel with multiple threads/processes.
Benefits for visualization

• See potential bottlenecks:
  • Where parallelism may not be possible
  • Areas where many tasks depend on each other,
  • May cause a great deal of communication overhead

Gives you an opportunity to tune and optimize your algorithm.
Dynamic Schedulers

• Executes the task graph

• The dask library contains schedulers for single-threaded, multi-threaded, multi-process, and distributed execution.

• Dask graphs are completely separate from the choice of scheduler

• All operate dynamically, meaning that execution order is determined during execution

• Steps:
  1. A worker reports that it has completed a task and that it is ready for another
  2. Update runtime state to record the finished task
  3. Choose a task to give to worker from among the set of ready-to-run tasks.
• Because these are out of core computations, we need to keep a small memory footprint
• Always want to optimize performance
• How do we select which tasks run first?
• Last in, first out is most effective. Select tasks whose data dependencies were most recently made available. Forces the scheduler to finish related tasks before starting new ones.
How does Dask work with other frameworks?

- Dask does have a machine learning library dask-ml
- Allows work with existing libraries like Scikit-Learn and XGBoost.
- Dask -> (joblib) -> Scikit-Learn
- Skorch allows PyTorch models to be wrapped in Scikit-learn compatible estimators
- The package SciKeras brings a Scikit-learn API to Keras

https://ml.dask.org/
Dask Delayed

• Sometimes problems might not need to use Dask Arrays or DataFrames
• Allows you to parallelize custom algorithms
• The delayed function decorates your functions so that they operate lazily
• Lazily functions do not execute immediately, only when you tell them to
• They are only added to a task graph
• Use compute() to start execution
```python
import dask
@dask.delayed
def sub3(a):
    return a - 3
@dask.delayed
def add3(a):
    return a + 3
def main():
    x, y = 0, 0
    for i in range(2):
        x = sub3(x)
        y = add3(y)
    total = x + y
    print(total.compute())

# Visualizing Task Graph
total.visualize()
```
Dask Futures

• Able to launch parallel tasks asynchronously in the background
• Extends Python’s concurrent.futures interface
• Very simple to use
• Not like delayed, executes immediately
• Dask automatically handles dependency tracking
• Tasks can launch other tasks by getting their own client
from dask.distributed import Client
import dask.array as da

def sub3(a):
    return a - 3

def add3(a):
    return a + 3

def main():
    client = Client()
    x = da.ones((10, 10), chunks=(10, 5))
    y = da.random.random((10, 10), chunks=(10, 5))
    futureX = client.submit(sub3, x)  # background thread or process
    futureY = client.submit(add3, y)

    total = futureX.result() + futureY.result()
    # Visualizing Task Graph
    total.visualize()
Dask Bag

• A bag is an unordered collection with repeats.

• Bounded by the Global Interpreter Lock, recommend switch from using a multi-threaded scheduler to a multi-processing one.

• Implements operations like map, filter, fold, and groupby. Follows PyToolz Api

• It combines the streaming and performance of projects like cytoolz with parallelism

• Useful in parsing and cleaning up initial data dumps like JSON or log files
Performance Limitations

• Scheduler spends a few microseconds on every task. There can be hundreds of thousands of tasks to do. For best performance, each task should take at least 10ms

• When Dask assigns tasks to the workers, it usually makes the right decision with heuristics. But non-optimal performance might occur

• It has all the limitations of Python

• Workers can cease functioning
  • An unrecoverable exception happens within the worker
  • The worker process is shut down by some external action

• Dask may run your function multiple times
Low Barrier Entry

• You can use your laptop to perform parallel analysis of datasets larger than memory.

• Provides same interface as NumPy and Pandas

• Users familiar with SciKit-Learn should be able to work with Dask-ML easily

• You can create complex algorithms without having to simultaneously develop expertise in a complex parallel programming framework.

• Dask is not oriented towards a specific problem domain, so there are a large amount of use cases.

• https://stories.dask.org/en/latest/
More

- https://docs.dask.org/en/latest/
- https://towardsdatascience.com/why-every-data-scientist-should-use-dask-81b2b850e15b
- https://www.youtube.com/watch?v=1kkFZ4P-XHg
- https://www.youtube.com/watch?v=RA_2qdipVng&list=PLRtz5iA93T4PQvWuoMnlyElz1fXiJ5Pri