Introduction to Dask bags

PARALLEL PROGRAMMING WITH DASK IN PYTHON

James Fulton
Climate Informatics Researcher

datacamp

What is unstructured data?

```
# Unstructured text data
string_list = [
    "Really good service ...",
    "This is the second time we've stayed ...",
    "Great older hotel. My husband took ...",
    ...]

# Semi-structured dictionary data
dict_list = [
    {"name": "Beth", "employment": [{"role": "manager", "start_date": ...}, ...]},
    {"name": "Omar", "employment": [{"role": "analyst", "start_date": ...}, ...]},
    {"name": "Fang", "employment": [{"role": "engineer", "start_date": ...}, ...]},
    ...]
```

Dask bags

```
import dask.bag as db

# Create Dask bag from list
bag_example = db.from_sequence(string_list, npartitions=5)
print(bag_example)
```

```
dask.bag<from_sequence, npartitions=5>
```

```
# Print single element from bag
print(bag_example.take(1))
```

```
('Really good service ...',)
```

Dask bags

```
import dask.bag as db

# Create Dask bag from list
bag_example = db.from_sequence(string_list, npartitions=5)
print(bag_example)
```

```
dask.bag<from_sequence, npartitions=5>
```

```
# Print two elements from bag
print(bag_example.take(2))
```

```
('Really good service ...', 'This is the second time we've stayed ...'',)
```

Number of elements

```
number_of_elements = bag_example.count()
print(number_of_elements)
```

```
<dask.bag.core.Item at ...>
```

```
print(number_of_elements.compute())
```

```
20491
```

Loading in text data

```
import glob
filenames = glob.glob('data/*.txt')
print(filenames)
```

```
["data/file_0.txt", "data/file_1.txt", "data/file_2.txt"]
```

```
text_data_bag = db.read_text(filenames)
```

```
text_data_bag = db.read_text('data/*.txt')
```

```
print(text_data_bag)
```

```
dask.bag<bag-from-delayed, npartitions=3>
```

String operations

```
text_data_bag = db.read_text('data/*.txt')


print(text_data_bag.take(1))
```

```
('Really good service ...',)
```

```
# Convert the text to upper case
print(text_data_bag.str.lower().take(1))
```

```
('really good service ...',)
```

String operations

```
# Change 'good' to 'great' in all places
print(text_data_bag.str.replace('good', 'great').take(1))
```

```
('Really great service ...',)
```

```
# How many times does 'great' appear the first 3 elements of the bag?
print(text_data_bag.str.count('great').take(3))
```

```
(0,1,5,)
```

Let's practice!

PARALLEL PROGRAMMING WITH DASK IN PYTHON

Dask bag operations

PARALLEL PROGRAMMING WITH DASK IN PYTHON

James Fulton
Climate Informatics Researcher

The map method

```
def number_of_words(s):
    word_list = s.split(' ')
    return len(word_list)

print(number_of_words('these are four words'))
```

```
4
```

The map method

```
string_list  = [
  'these are four words',
  'but these are five words',
  'and these are seven words in total'
]
# Create bag from list above
string_bag = db.from_sequence(string_list)

# Apply function to each element in bag
word_count_bag = string_bag.map(number_of_words)

print(word_count_bag)
```

```
dask.bag<number_of_words, npartitions=3>
```

The map method

```
string_list  = [
  'these are four words',
  'but these are five words',
  'and these are seven words in total'
]
# Create bag from list above
string_bag = db.from_sequence(string_list)

# Apply function to each element in bag
word_count_bag = string_bag.map(number_of_words)
# Run compute method
print(word_count_bag.compute())
```

```
[4, 5, 7]
```

JSON data

- Inside an example JSON file, `example_0.json`

```
{"name": "Beth", "employment": [{"role": "manager", "start_date": ...}, ...], ...}
{"name": "Omar", "employment": [{"role": "analyst", "start_date": ...}, ...], ...}
{"name": "Fang", "employment": [{"role": "engineer", "start_date": ...}, ...], ...}
...
```

```
text_bag = db.read_text('example*.json')
print(text_bag.take(1))
```

```
('{"name": "Beth", "employment": [{"role": "manager", "start_date": ...}, ...]}\n',)
```

- This is just a string

Converting JSON from string to dictionary

```
import json

text_bag = db.read_text('example*.json')

dict_bag = text_bag.map(json.loads)
print(dict_bag.take(1))
```

```
({"name": "Beth",
  "employment": [{"role": "manager", "start_date": ...}, ...]
  ...},)
```

- Now this is a Python dictionary

Filtering

```
def is_new(employee_dict):
    """Check if employee has less than 1 years service"""
    return employee_dict['years_service'] < 1

# Select only the newer employees
new_employee_bag = dict_bag.filter(is_new)
```

```
# Count all employees and new employees
print(dict_bag.count().compute(), new_employee_bag.count().compute())
```

```
261 49
```

Filtering

We can use a lambda function to do the same thing

```
def is_new(employee_dict):
    """Check if employee has less than 1 years service"""
    return employee_dict['years_service'] < 1

# Can use a lambda function instead of writing the function above
new_employee_bag = dict_bag.filter(lambda x: x['years_service'] < 1)
```

Pluck method

- Inside an example JSON file, `example_0.json`

```
{"name": "Beth", "employment": [{"role": "manager", "start_date": ...}, ...], ...}
{"name": "Omar", "employment": [{"role": "analyst", "start_date": ...}, ...], ...}
{"name": "Fang", "employment": [{"role": "engineer", "start_date": ...}, ...], ...}
...
```

```
employment_bag = new_employee_bag.pluck('employment')
print(employment_bag.take(1))
```

```
([{"role": "manager", "start_date": ...}, ...],)
```

Pluck method

```
employment_bag = new_employee_bag.pluck('employment')

number_of_jobs_bag = employment_bag.map(len)

print(number_of_jobs_bag.take(1))
```

```
(4,)
```

Aggregations

```
min_jobs = number_of_jobs_bag.min()
max_jobs = number_of_jobs_bag.max()
mean_jobs = number_of_jobs_bag.mean()

print(dask.compute(min_jobs, max_jobs, mean_jobs))
```

```
(0, 12, 3.142)
```

Let's practice!

PARALLEL PROGRAMMING WITH DASK IN PYTHON

Converting unstructured data to a DataFrame

PARALLEL PROGRAMMING WITH DASK IN PYTHON

James Fulton
Climate Informatics Researcher

Nested JSON data

- Inside an example JSON file, `example_0.json`

```
{"name": "Beth", "employment": [{"role": "manager", "start_date": ...}, ...], ...}
{"name": "Omar", "employment": [{"role": "analyst", "start_date": ...}, ...], ...}
{"name": "Fang", "employment": [{"role": "engineer", "start_date": ...}, ...], ...}
...
```

Restructuring a dictionary

```
def add_number_of_jobs(employee_dict):
    employee_dict['number_of_previous_jobs'] = len(employee_dict['employment'])
    return employee_dict

dict_bag = dict_bag.map(add_number_of_jobs)
```

Removing parts of the dictionary

```
def delete_dictionary_entry(dictionary, key_to_drop):
    del dictionary[key_to_drop]
    return dictionary

dict_bag = dict_bag.map(delete_dictionary_entry, key_to_drop='employment')
```

Selecting parts of the dictionary

```
def filter_dictionary(dictionary, keys_to_keep):
    new_dict = {}
    for k in keys_to_keep:
        new_dict[k] = dictionary[k]
    return new_dict

dict_bag = dict_bag.map(
    filter_dictionary,
    keys_to_keep=['name', 'number_of_previous_jobs']
)
```

Converting to DataFrame

```
print(dict_bag.take(1))
```

```
({'name': 'Beth',
  'number_of_previous_jobs': 3},)
```

```
converted_bag_df = dict_bag.to_dataframe()
print(converted_bag_df)
```

```
                name    number_of_previous_jobs
npartitions=3
               object                    float64
                  ...                        ...
```

Let's practice!

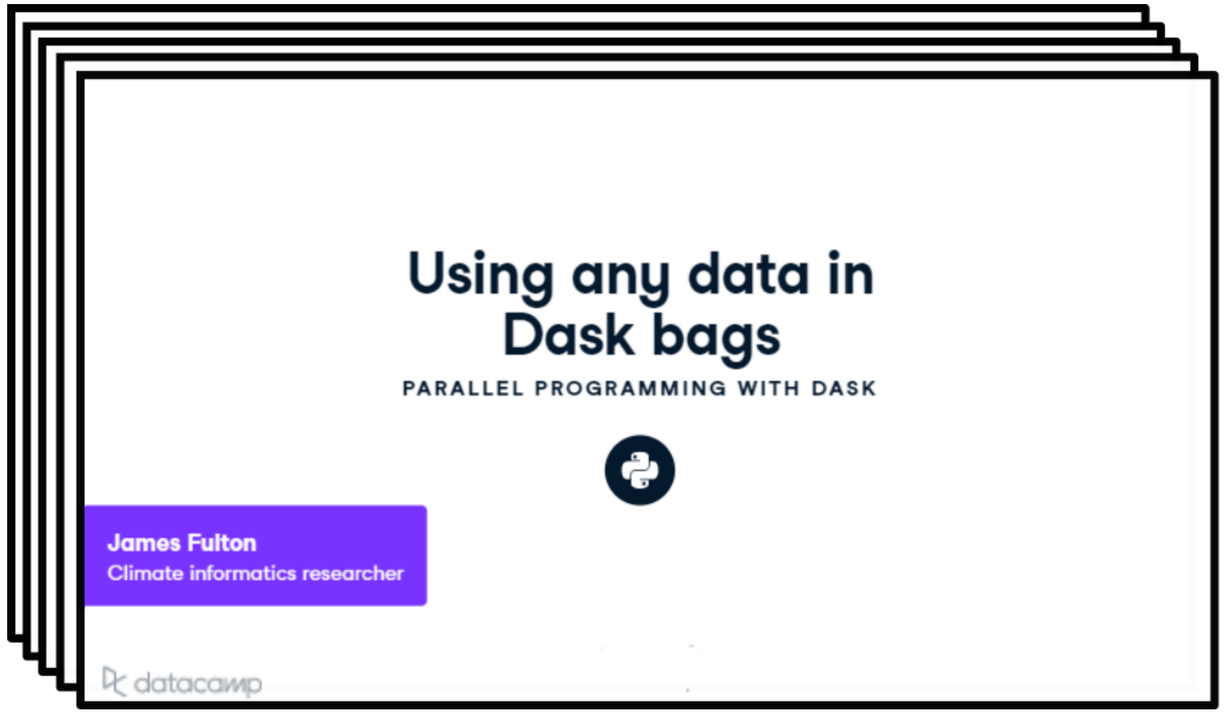
PARALLEL PROGRAMMING WITH DASK IN PYTHON

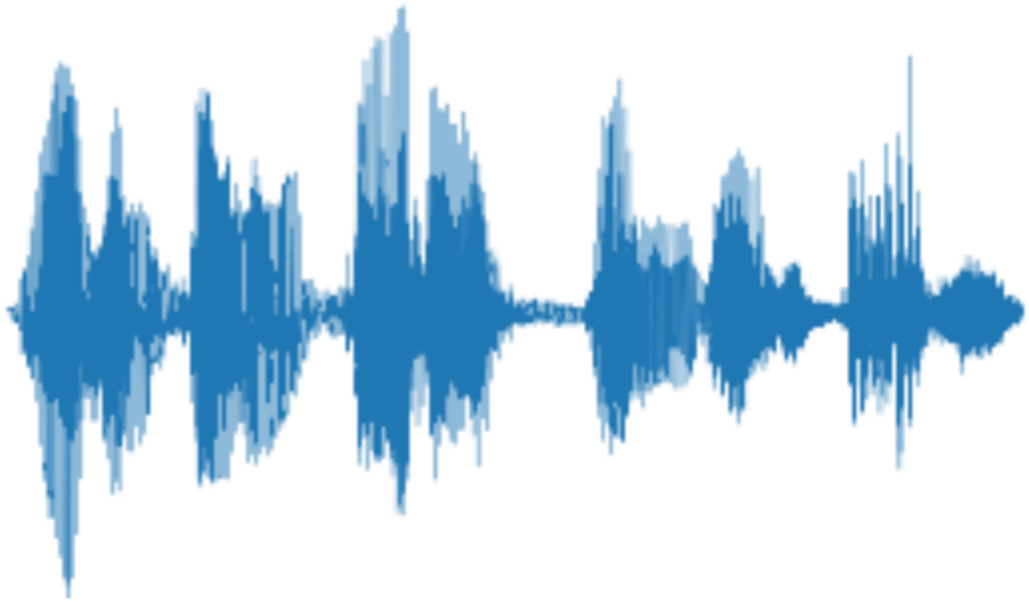
Using any data in Dask bags

PARALLEL PROGRAMMING WITH DASK IN PYTHON

James Fulton
Climate Informatics Researcher

Complex mixed data formats





Creating a Dask bag

```
import glob


video_filenames = glob.glob("*.mp4")
```

```
print(video_filenames)
```

```
['me_at_the_zoo.mp4', 'life_goes_on.mp4', 'guitar.mp4', 'hurt.mp4', ...]
```

Creating a Dask bag

```
import glob

video_filenames = glob.glob("*.mp4")
```

```
import dask.bag as db

filename_bag = db.from_sequence(video_filenames)
```

```
filename_bag.take(1)[0]
```

```
'me_at_the_zoo.mp4'
```

Loading custom data

```
# Loads a single video
load_mp4("video.mp4")
```

```
{'video': array(
         [[[ 51,  57,  37, ..., 227, 238, 168],
         ...,
         [ 83, 125, 129, ..., 222, 148, 208]]]),
 'audio': array([   7. ,    9. ,    9.5, ..., -544.5, -551. , -558. ]),
 'filename': 'video.mp4'}
```

Loading custom data

```
data_bag = filename_bag.map(load_mp4)
```

```
data_bag.take(1)[0]
```

```
{'video': array(
         [126, 162, 203, ...,  63,  58,   8],
         ...,
         [ 58, 222, 170, ..., 234,  63,  81]]]),
 'audio': array([-203.5, -209. , -207. , ..., -222.5, -233. , -248.5]),
 'filename': 'me_at_the_zoo.mp4'}
```

Loading custom data

```
data_bag = filename_bag.map(load_mp4)
```

```
# Create empty list
data_list = []

# Add delayed loaded files to list
for file in video_filenames:
    data_list.append(dask.delayed(load_mp4)(file))
```

List of delayed objects vs. Dask bag

```
# Convert list of delayed objects to dask bag
data_bag = db.from_delayed(data_list)
```

```
# Convert dask bag to list of delayed objects
data_list = data_bag.to_delayed()
```

Further analysis

```
transcribed_bag = data_bag.map(transcribe_audio)
```

```
transcribed_bag.take(1)[0]
```

```
{'video': array(
         [126, 162, 203, ...,  63,  58,   8],
         ...,
         [ 58, 222, 170, ..., 234,  63,  81]]]),
 'audio': array([-203.5, -209. , -207. , ..., -222.5, -233. , -248.5]),
 'filename': 'me_at_the_zoo.mp4'
 'transcript': "All right, so here we are in front of the, uh, elephants ...",
}
```

Further analysis

```
# Apply custom function to remove videos with no spoken words
clean_bag = transcribed_bag.filter(transcript_is_not_blank)

# Apply sentiment analysis to transcripts
sentiment_bag = clean_bag.map(analyze_transcript_sentiment)
```

```
# Remove unwanted elements from bag
keys_to_drop = ['video', 'audio']
final_bag = sentiment_bag.map(filter_dictionary, keys_to_drop=keys_to_drop)

# Convert to Dask DataFrame
df = final_bag.to_dataframe()
```

Results

```
df.compute()
```

```
            filename                transcript         sentiment
0  me_at_the_zoo.mp4  All right, so here ...          positive
...              ...                     ...               ...
```

Using .wav files

```
# Import scipy module for .wav files
from scipy.io import wavfile

# Load sampling frequency and audio array
sample_freq, audio = wavfile.read(filename)
```

Using .wav files

```
# Samples per second
print(sample_freq)
```

```
44100
```

```
# The audio data
print(audio)
```

```
array([ 148,  142,  150, ..., -542, -546, -559], dtype=int16)
```

Let's practice!

PARALLEL PROGRAMMING WITH DASK IN PYTHON