dask-elk Documentation

Release 0.1.0

Apostolos Vlachopoulos

Apr 14, 2020

Documentation

Dask-ELK tries to immitate the ES-Hadoop functionality and connect with an Elasticsearch cluster.

CHAPTER 1

Install Dask-ELK

You can install via `pip`

1.1 Pip

You can istall dask via pip with the following command:: pip install dask-elk

CHAPTER 2

Dask-ELK usage

In order to user `dask-elk`, first you need to create an instance of the DaskElasticClient

To connect to an Elasticsearch cluster in localhost:

```
from dask_elk.client import DaskElasticClient
client = DaskElasticClient(host='localhost')
```

2.1 Read dataframe

To read from data from the elasticsearch:

```
my_index ='myindex'
df = client.read(index=my_index, doc_type='_doc')
```

Client will do the following work:

- Identify the publish address of the cluster's nodes
- **Obtain information on the index:**
  - Mapping of the index
  - In how many shards is index divided
  - Which nodes contain the shards of the index

By default client will try to connect to each node containing the shards and issue the request on each node. Thus it will create number of tasks equal to the number of shards per requested index You can increase the number of tasks by setting the number of documents per partition:

```
df = client.read(index=my_index, doc_type='_doc', number_of_docs_per_partition=1000)
```

The returned object is a dask dataframe.

You can also specify a dsl query to be pushed down to Elasticsearch to minimize the data returned back

```
df = client.read(query=query, index=my_index, doc_type='_doc')
```

The above code will push down the query to the Elasticsearch and minimize the returned data. Since `DaskElasticClient` uses the scroll api from Elasticsearch, aggregation queries are not supported.

2.2 Save dataframe

You can save your data back to an Elasticsearch index:

```
df = client.save(df, index='index_to_save', doc_type='_doc', action='index')
```

Save method uses bulk actions under the hood to save documents back to elasticsearch. The number of tasks depends on the number of paritions of the passed `df`

You can use some python fromat functions to dynamically create indices during save (The operation is experimental) e.g assuming that timestamp is a datetime column then the folloing code will try to index documents on idices splited per day

```
index = 'index-{timestamp:%Y.%m.%d}'
df = client.save(df, index=index, doc_type='_doc', action='index')
```

To update documents provided `df` needs to contain `_id` column. You should also provide `update` as action parameter

CHAPTER 3

DaskElasticSearch API

3.1 DaskElasticSearch

DaskElasticClient([host, port, ...])	Basic object.
DaskElasticClient.read([query, index, ...])	Method to read from elasticsearch index/ices
DaskElasticClient.save(data, index, doc_type)	Save a dask dataframe to Elasticsearch

3.2 Read Dataframe

class dask_elk.client.**DaskElasticClient**(*host='localhost'*, *port=9200*, *client_klass=<class 'elasticsearch.client.Elasticsearch'>*, *username=None*, *password=None*, *wan_only=False*, ***client_kwargs*)

Basic object. Client to connect with Elasticsearch

Parameters

- **host** (*list[str]|str*) – The host to connect to
- **port** (*int*) – The port to connect to
- **client_klass** (*elasticsearch.Elasticsearch*) – The class to use to create the Elasticsearch client instance
- **username** (*str|None*) – The username if X-pack is activated
- **password** (*str|None*) – The password if X-pack is activated
- **wan_only** (*bool*) – If client should perform Node lookup. Set to True if ELK behind firewall or if nodes not accessed directly
- **client_kwargs** (*dict[str,T]*) – Arguments to pass to Elasticsearch client object created by client_klass

read(*query=None*, *index=None*, *doc_type=None*, *number_of_docs_per_partition=1000000*, *size=1000*, *fields_as_list=None*, ***kwargs*)
Method to read from elasticsearch index/ices

Parameters

- **query** (*dict(str, T)|None*) – The query to push down to ELK.
- **index** (*str*) – String of the index/ices to execute query on.
- **doc_type** (*str*) – Type index belongs too
- **number_of_docs_per_partition** (*int*) – Number of documents for each partition/task created by the readers
- **size** (*int*) – The scroll size
- **fields_as_list** (*str|None*) – Comma separated list of fields to be treated as object
- **kwargs** – Additional keyword arguments to pass to the search method of python Elasticsearch client

Returns Dask Dataframe containing the data

Return type dask.dataframe.DataFrame

save(*data*, *index*, *doc_type*, *action='index'*)
Save a dask dataframe to Elasticsearch

Parameters

- **data** (*dask.dataframe.DataFrame*) – Dataframe to save into ELK
- **index** (*str*) – The index to save dataframe
- **doc_type** (*str*) – Index doc type
- **action** (*str*) – index if indexing you data 'update' if updating data

Returns The data with the save applied on it

Return type dask.dataframe.DataFrame

CHAPTER 4

CHANGELOG

4.1 0.4.0 -2020-04-14

4.1.1 Fix

- Better handling of dataframes with numpy arrays as column value (#24 frascuchon)

4.2 0.3.0 -2020-01-17

4.2.1 Fix

- Compatibility with newer elasticsearch versions >= 6.8.0 (#23 frascuchon)

4.3 0.2.0 - 2019-02-17

4.3.1 Core

- Dropped support python version 2.x

4.3.2 Documentation

- Added Changelog

Index

D

R

S