ETL process with TURBODBC

Reasoning

In this demo we will upload data to a SQL Server database using *TURBODBC*.

The principal reason for turbodbc is: for uploading real data, *pandas.to_sql* is painful slow, and the workarounds to make it better are pretty hairy, if you ask me. After many hours running in circles around pandas workarounds, I gave up on it, but just because I discovered *TURBODBC*, this piece of pure love!

Get TURBODBC on https://turbodbc.readthedocs.io/en/latest/index.html

Comparison

Quick comparison: in this script, for loading 10000 lines, 77 columns, we have:

- *pandas.to_sql* took almost 200 seconds to finish
- *turbodbc* took only 3 seconds…

Step by step summary:

In this python script, we will:

- create a *sqlAlchemy* connection to our database in a SQL Server
- load and treat some data (in my case, a DataFrame containing 77 columns, 350k+ lines)
- upload it to our SQL Server using *pandas.to_sql*
- create a *turbodbc* connection
- upload the same sample of data, but this time using *turbodbc*
- profit!

Tools used:

- Python 3.6.7 :: Anaconda, Inc.
- TURBODBC version '3.0.0'
- sqlAlchemy version '1.2.12'
- pandas version '0.23.4'

The code

The imports

```
import sqlalchemy
import pandas as pd
import numpy as np
import turbodbc
import credenciais
import time
```

Create the table using sqlAlchemy

Create connection

```
mydb = 'someDB'


def make_con(db):
    """Connect to a specified db."""
    database_connection = sqlalchemy.create_engine(
        'mssql+pymssql://{0}:{1}@{2}/{3}'.format(
            credenciais.myuser, credenciais.mypassword,
            credenciais.myhost, db
            )
        )
    return database_connection


pd_connection = credenciais.make_con(mydb)
```

Load and treat some data

Substitute my sample.pkl for yours:

```
df = pd.read_pickle('sample.pkl')

df.columns = df.columns.str.strip()
df = df.applymap(str.strip)
df = df.replace('', np.nan)
df = df.dropna(how='all', axis=0)
df = df.dropna(how='all', axis=1)
df = df.replace(np.nan, 'NA')  # turbodbc hates null values...
df.shape
```

Create table

Using pandas + sqlAlchemy, but just for preparing room for turbodbc:

```
table = 'testing'
df.head().to_sql(table, con=pd_connection, index=False)
```

Turbodbc connection

```
connection = turbodbc.connect(
                                driver="SQL Server",
                                server=credenciais.myhost,
                                database=mydb,
                                uid=credenciais.myuser,
                                pwd=credenciais.mypassword
                            )
```

Preparing sql comands and data for turbodbc

```
def turbo_write(mydb, df, table):
    """Use turbodbc to insert data into sql."""
    start = time.time()
    # preparing columns
    colunas = '('
    colunas += ', '.join(df.columns)
    colunas += ')'

    # preparing value place holders
    val_place_holder = ['?' for col in df.columns]
    sql_val = '('
    sql_val += ', '.join(val_place_holder)
    sql_val += ')'

    # writing sql query for turbodbc
    sql = f"""
    INSERT INTO {mydb}.dbo.{table} {colunas}
    VALUES {sql_val}
    """

    # writing array of values for turbodbc
    valores_df = [df[col].values for col in df.columns]

    # cleans the previous head insert
    with connection.cursor() as cursor:
        cursor.execute(f"delete from {mydb}.dbo.{table}")
        connection.commit()

    # inserts data, for real
    with connection.cursor() as cursor:
        try:
            cursor.executemanycolumns(sql, valores_df)
            connection.commit()
        except Exception:
            connection.rollback()
            print('something went wrong')

    stop = time.time() - start
    return print(f'finished in {stop} seconds')
```

Writes data using turbodbc

I've got 10000 lines (77 columns) in 3 seconds:

```
turbo_write(mydb, df.sample(10000), table)
```

Pandas method comparison

I've got the same 10000 lines (77 columns) in 198 seconds…

```
table = 'pd_testing'


def pandas_comparisson(df, table):
    """Load data using pandas."""
    start = time.time()
    df.to_sql(table, con=pd_connection, index=False)
    stop = time.time() - start
    return print(f'finished in {stop} seconds')


pandas_comparisson(df.sample(10000), table)
```

The autor

Written on 2018-11-07 by *Erick Gomes Anastácio*

Data Scientist, physicist, living in São Paulo, Brazil.

Senior Consultant at Control Risks

erickfis@gmail.com
https://erickfis.github.io/portfolio/
https://www.linkedin.com/in/erick-anastácio-15241717/