GPU Accelerated DataFrames in Python

Getting Started Cheat Sheet

CREATE

Instantiate DataFrames from files and host memory.

Create a DataFrame.

cudf.DataFrame([1,2,3,4], columns=['foo']) - from a list of elements

cudf.DataFrame({'foo': [1,2,3,4], 'bar': ['a','b','c',None]}) - from a dictionary of columns

cudf.DataFrame([(1,'a'), (2,'b')], columns=['foo','bar']) - from a list of tuples

cudf.from_pandas(pd.DataFrame([1,2,3,4], columns=['ints'])) - Convert pandas DataFrame (CPU) to cuDF DataFrame (GPU).

cudf.read_csv('results.csv') - Read contents of a CSV file.

cudf.read_csv('results.csv', nrows=2, usecols=['foo']) - Read two rows and column foo of a CSV file.

cudf.read_csv('results.csv', skiprows=1, names=['foo','bar']) - Replace column names when reading a CSV file.

cudf.read_json('results.json') - Read contents of a JSON file.

cudf.read_json('results.json', lines=True, engine='cudf') - Read contents of lines-formatted JSON file using GPU.

cudf.read_parquet('results/df_default.parquet') - Read contents of a Parquet file.

cudf.read_parquet('results/df_default.parquet', columns=['foo']) - Read column foo from a Parquet file.

Create a series.

cudf.Series([0,1,2,3]) - from a list of elements

df['foo'] - get column 'foo' from DataFrame as a cuDF Series

PROPERTIES

Extract properties from DataFrames and Series

FOR DATAFRAMES

df.columns - Get a list of column names.

df.dtypes - Get a list of columns with data types.

Retrieve rows and columns by index label.

df.loc[3] - row with index 3

df.loc[3, 'foo'] - row with index 3 and column 'foo'

df.loc[2:5, ['foo', 'bar']] - rows with labels 2 to 5 and columns 'foo' and 'bar'

df.shape - Know data shape (row #, col #)

df.size - Know total number of elements.

df.values - Get an array with all elements.

PROPERTIES

Extract properties from DataFrames and Series

FOR SERIES

Retrieve rows and columns by index label.

ser.loc[1] - row with index 1

ser.loc[1:4] - row with indicies 1 to 4

ser.values - Get an array of all elements.

SAVE

Persist data to disk or convert to other memory representations.

df.to_csv('results.csv') - Save cuDF DataFrame in a CSV format with index and header.

df.to_csv('results.csv', index=False, header=False) - Save cuDF DataFrame in a CSV format without index and header.

df.to_dlpack() - Convert DataFrame to DLPack tensor for deep learning.

df.to_json('results.json') - Save cuDF DataFrame in a JSON format.

df.to_json('results.json', orient='records', lines=True) - Save cuDF DataFrame in a JSON Lines format.

df.to_pandas() - Convert cuDF DataFrame (GPU) to pandas DataFrame (CPU).

df.to_parquet('results.parquet') - Save cuDF DataFrame in a Parquet format.

QUERY

Extract information from data.

df.head() - Retrieve top 5 rows from DataFrame.

df.head(2) - Retrieve top 2 rows from DataFrame.

df.memory_usage() - Learn how much memory your DataFrame consumes (in bytes).

df.nlargest(3, 'foo') - Retrieve 3 rows with largest values in column foo.

df.nsmallest(2, 'foo') - Retrieve 2 rows with smallest values in column foo.

df.query('foo == 1') - Get all rows where column foo equals to 1.

df.query('foo > 10') - Get all rows where column foo is greater than 10.

df.sample() - Fetch a random row.

df.sample(3) - Fetch a random 3 rows.

TRANSFORM

Alter the information and structure of DataFrames

df.apply_rows(func, incols=['foo'], outcols={'bar': 'float64'}, kwargs={}) - Apply custom transformation defined in func to column foo and store in column bar.

TRANSFORM

Alter the information and structure of DataFrames

```
def func(foo, bar):
    for i, f in enumerate(foo):
        bar[i] = f + 1
```
- Kernel definition to use in apply_rows() function.

cudf.concat([df1, df2]) - Append a DataFrame to another DataFrame.

df.drop(1) - Remove row with index equal to 1.

df.drop([1,2]) - Remove rows with index equal to 1 and 2.

df.drop('foo', axis=1) - Remove column foo.

df.dropna() - Remove rows with one or more missing values.

df.dropna(subset='foo') - Remove rows with a missing value in column foo.

df.fillna(-1) - Replace any missing value with a default.

df.fillna({'foo': -1}) - Replace a missing value in column foo with a default.

df1.join(df2) - Join with a DataFrame on index.

df1.merge(df2, on='foo', how='inner') - Perform an inner join with a DataFrame on column foo.

df1.merge(df2, left_on='foo', right_on='bar', how='left') - Perform a left outer join with a DataFrame on different keys.

df.rename({'foo': 'bar'}, axis=1) - Rename column foo to bar.

df.rename({1: 101}) - Replace index 1 with value 101.

df.reset_index() - Replace index and retain the old one as a column.

df.reset_index(drop=True) - Replace index and remove the old one.

df.set_index('foo') - Replace index with the values of column foo.

df.set_index('foo', drop =False) - Replace index with the values of column foo and retain the column.

SUMMARIZE

Learn from data by aggregating and exploring.

df.groupby(by='foo').agg({'bar': 'sum', 'baz': 'count'}) - Aggregate DataFrame: sum elements of bar, count elements of baz by values of foo.

df.describe() - Learn basic statistics about DataFrame.

df.describe(percentiles=[.1,.9]) - Learn basic statistics about DataFrame and only produce 1st and 9th decile.

df.max() - Learn the maximum value in each column.

df.max(axis=1) - Learn the maximum value in each row.

df.mean() - Learn the average value of each column.

df.mean(axis=1) - Learn the average value of each row.

df.min() - Learn the minimum value in each column.

df.min(axis=1) - Learn the minimum value in each row.

df.quantile() - Learn the median of each column.

df.quantile(.25) - Learn the 1st quartile of each column.

df.std() - Learn the standard deviation of each column.

df.std(axis=1) - Learn the standard deviation of each row.

df.sum() - Get the sum of each column.

df.sum(axis=1) - Get the sum of each row.

ser.unique() - Find all unique values in Series.

STRING

Operate on string columns on GPU.

ser.str.contains('foo') - Check if Series of strings contains foo.

ser.str.contains('foo[a-z]+') - Check if Series of strings contains words starting with foo.

ser.str.extract('(foo)') - Retrieve regex groups matching pattern in Series of strings.

ser.str.extract('[a-z]+flow (\d)') - Retrieve IDs of dataflows, workflows, etc., in Series of strings.

ser.str.findall('([a-z]+flow)') - Retrieve all instances of words like dataflow, workflow, etc.

ser.str.len() - Find the total length of a string.

ser.str.lower() - Cast all the letters in a string to lowercase characters.

ser.str.match('[a-z]+flow') - Check if every element matches the pattern.

ser.str.ngrams_tokenize(n=2, separator='_') - Generate all bi-grams from a string separated by underscore.

ser.str.pad(width=10) - Make every string of equal length.

ser.str.pad(width=10, side='both', fillchar='$') - Make every string of equal length with word centered and padded with dollar signs.

ser.str.replace('foo', 'bar') - Replace all instances of word foo with bar.

ser.str.replace('f..', 'bar') - Replace all instances of 3-letter words beginning with f with bar.

ser.str.split() - Split the string on spaces.

ser.str.split(',', n=5) - Split the string on comma and retain only the first 5 occurences (6 column retains the remainder of the string).

tokens, masks, metadata = ser.str.subword_tokenize('hash.txt') - Tokenize text using perfectly hashed BERT vocabulary.

ser.str.upper() - Cast all the letters in a string to uppercase characters.

CATEGORICAL

Work with categorical columns on GPU.

ser.cat.add_categories(['foo','bar']) - Extend the list of categorical allowed values.

ser.cat.categories - Retrieve the list of all categories.

ser.cat.remove_categories(['foo']) - Remove the foo category from categorical column.

DATETIME

Deal with date and time columns on GPU.

ser.dt.day - Extract day from DateTime column.

ser.dt.dayofweek - Extract the day of a week from DataTime column.

ser.dt.year - Extract year from DateTime column.

MATH/STAT

Perform mathematical and statistical operations on columns.

df.corr() - Calculate coefficient of correlation.

df.exp() - Exponentiate values in all columns.

df.kurt() - Find kurtosis of each column.

df.log() - Take a logarithm of values in all columns.

df.pow(2) - Raise values in all columns to the power of 2.

df.skew() - Find skewness of each column.

df.sqrt() - Find root squares of values in all columns.