KDnuggets | Visit KDnuggets.com for more cheatsheets and additional learning resources.

RAPIDS cuDF Cheat Sheet

RAPIDS cuDF is an open-source Python library for GPU-accelerated DataFrames. cuDF provides a Pandas-like API that allows data engineers, analysts, and data engineers can use perform data manipulation and analysis tasks on large datasets and time series data using the power of NVIDIA GPUs allowing for faster data processing and analysis.

Getting started with cuDF is straightforward, especially if you have experience using Python and libraries like Pandas. While both cuDF and Pandas offer similar APIs for data manipulation, there are specific types of problems in which cuDF can provide significant performance improvements over Pandas, including large scale datasets, data preprocessing and engineering, real-time analytics, and, of course, parallel processing. The bigger the dataset, the greater the performance benefits.

Installation

Anaconda
$ conda create -n rapids-23.02 -c rapidsai -c conda-forge -c nvidia rapids=23.02 python=3.10 cudatoolkit=11.8

PIP Install
$ pip install cudf-cu11 dask-cudf-cu11 --extra-index-url=https://pypi.nvidia.com

>>> Refer docs.rapids.ai/install for latest instructions

Reading Dataset

The cupy is similar to NumPy API, which runs on GPU. We will import libraries and use cupy to generate random numbers to create cudf data frame.

```
import cupy,cudf

df = cudf.DataFrame(
  {
     "A": cupy.random.randint(5, 35, size=1000),
     "B": cupy.random.randint(300, 400, size=1000),
     "C": cupy.random.randint(1, 50, size=1000),
  }
)
df.head()
```

A	B	C
0	8	385
1	33	335
2	12	356
3	34	329
4	20	344

You can also read CSV files
df = cudf.read_csv("california_housing.csv")

Other formats
- cudf.read_json()
- cudf.read_text()
- cudf.read_parquet()
- cudf.read_hdf()
- cudf.read_feather()

Writing Dataframes

Saving CSV file
df.to_csv("processed_california_housing.csv")

Other formats
- df.to_json()
- df.to_parquet()
- df.to_hdf()
- df.to_feather()

Selection

Getting Data
df["B"].head()

```
0   331
1   363
2   350
3   364
4   358
```

Selection by Label
df.loc[2:5, ["A", "B"]]

Selection by Position
df.iloc[0:3, 0:2]

Boolean Indexing
df[df.A > 24]

Query
df.query("B == 344")

Missing Data

Find missing values
df.isna().sum()

Fill missing values
df.fillna(999)

Apply Function

```
def add_five(num):
   return num + 5

df["C"].apply(add_five)
```

Concat & Join

Concat
df_X = df.copy()

df_final = cudf.concat([df, df_X])

Join
df.merge(df_X, on=["B"], how="left")

Grouping

Grouping by B columns and summing
df.groupby("B").sum()

Grouping and applying statistical functions to specific columns
df.groupby("B").agg({"A": "max", "C": "mean"})

Converting Data Representation

df_pd = df.to_pandas()
df_numpy = df.to_numpy()
df_arrow = to_arrow()

Subscribe to KDnuggets News

Abid Ali Awan | 2023