storage_benchmark

September 29, 2021

[1]:
```
from pathlib import Path
import pandas as pd
import numpy as np
import random
import string
```

[2]:
```
results = {}
```

0.1 Generate Test Data

The test `DataFrame` that can be configured to contain numerical or text data, or both. For the HDF5 library, we test both the fixed and table format.

[3]:
```
def generate_test_data(nrows=100000, numerical_cols=2000, text_cols=0,␣
↪text_length=10):
    ncols = numerical_cols + text_cols
    s = "".join([random.choice(string.ascii_letters)
                 for _ in range(text_length)])
    data = pd.concat([pd.DataFrame(np.random.random(size=(nrows,␣
↪numerical_cols))),
                      pd.DataFrame(np.full(shape=(nrows, text_cols),␣
↪fill_value=s))],
                     axis=1, ignore_index=True)
    data.columns = [str(i) for i in data.columns]
    return data
```

[4]:
```
df = generate_test_data()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Columns: 2000 entries, 0 to 1999
dtypes: float64(2000)
memory usage: 1.5 GB
```

0.2 Parquet

0.2.1 Size

[7]:
```
parquet_file = Path('test.parquet')
```

[8]:
```
df.to_parquet(parquet_file)
size = parquet_file.stat().st_size
```

0.2.2 Read

[9]:
```
%%timeit -o
df = pd.read_parquet(parquet_file)
```

```
809 ms ± 14.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[9]:
```
<TimeitResult : 809 ms ± 14.2 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[10]:
```
read = _
```

[11]:
```
parquet_file.unlink()
```

0.2.3 Write

[12]:
```
%%timeit -o
df.to_parquet(parquet_file)
parquet_file.unlink()
```

```
12.1 s ± 32 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[12]:
```
<TimeitResult : 12.1 s ± 32 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[13]:
```
write = _
```

0.2.4 Results

[14]:
```
results['parquet'] = {'read': read.all_runs, 'write': write.all_runs, 'size':␣
↪size}
```

0.3 HDF5

[15]:
```
test_store = Path('index.h5')
```

0.3.1 Fixed Format

Size

[16]:
```
with pd.HDFStore(test_store) as store:
    store.put('file', df)
size = test_store.stat().st_size
```

Read

[17]:
```
%%timeit -o
with pd.HDFStore(test_store) as store:
    store.get('file')
```

```
449 ms ± 9.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[17]:
```
<TimeitResult : 449 ms ± 9.4 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[18]:
```
read = _
```

[19]:
```
test_store.unlink()
```

Write

[20]:
```
%%timeit -o
with pd.HDFStore(test_store) as store:
    store.put('file', df)
test_store.unlink()
```

```
4.2 s ± 28.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[20]:
```
<TimeitResult : 4.2 s ± 28.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[21]:
```
write = _
```

Results

[22]:
```
results['hdf_fixed'] = {'read': read.all_runs, 'write': write.all_runs, 'size':␣
↪size}
```

0.3.2 Table Format

Size

[23]:
```
with pd.HDFStore(test_store) as store:
    store.append('file', df, format='t')
size = test_store.stat().st_size
```

Read

[24]:
```
%%timeit -o
with pd.HDFStore(test_store) as store:
```

```
    df = store.get('file')
```

```
508 ms ± 9.14 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[24]:
```
<TimeitResult : 508 ms ± 9.14 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[25]:
```
read = _
```

[26]:
```
test_store.unlink()
```

Write Note that `write` in table format does not work with text data.

[27]:
```
%%timeit -o
with pd.HDFStore(test_store) as store:
    store.append('file', df, format='t')
test_store.unlink()
```

```
2.92 s ± 11.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[27]:
```
<TimeitResult : 2.92 s ± 11.6 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[28]:
```
write = _
```

Results

[29]:
```
results['hdf_table'] = {'read': read.all_runs, 'write': write.all_runs, 'size':␣
↪size}
```

0.3.3 Table Select

Size

[30]:
```
with pd.HDFStore(test_store) as store:
    store.append('file', df, format='t', data_columns=['company', 'form'])
size = test_store.stat().st_size
```

Read

[31]:
```
company = 'APPLE INC'
```

[32]:
```
# %%timeit
# with pd.HDFStore(test_store) as store:
#     s = store.select('file', 'company = company')
```

[33]:
```
# read = _
```

[34]:
```
# test_store.unlink()
```

Write

[35]:
```
# %%timeit
# with pd.HDFStore(test_store) as store:
#     store.append('file', df, format='t', data_columns=['company', 'form'])
# test_store.unlink()
```

[36]:
```
# write = _
```

Results

[37]:
```
# results['hdf_select'] = {'read': read.all_runs, 'write': write.all_runs,␣
↪'size': size}
```

0.4 CSV

[38]:
```
test_csv = Path('test.csv')
```

0.4.1 Size

[39]:
```
df.to_csv(test_csv)
test_csv.stat().st_size
```

[39]:
```
3854558165
```

0.4.2 Read

[40]:
```
%%timeit -o
df = pd.read_csv(test_csv)
```

```
31.5 s ± 163 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[40]:
```
<TimeitResult : 31.5 s ± 163 ms per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

[41]:
```
read = _
```

[42]:
```
test_csv.unlink()
```

0.4.3 Write

[43]:
```
%%timeit -o
df.to_csv(test_csv)
test_csv.unlink()
```

```
3min 7s ± 1.95 s per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

[43]:
```
<TimeitResult : 3min 7s ± 1.95 s per loop (mean ± std. dev. of 7 runs, 1 loop
each)>
```

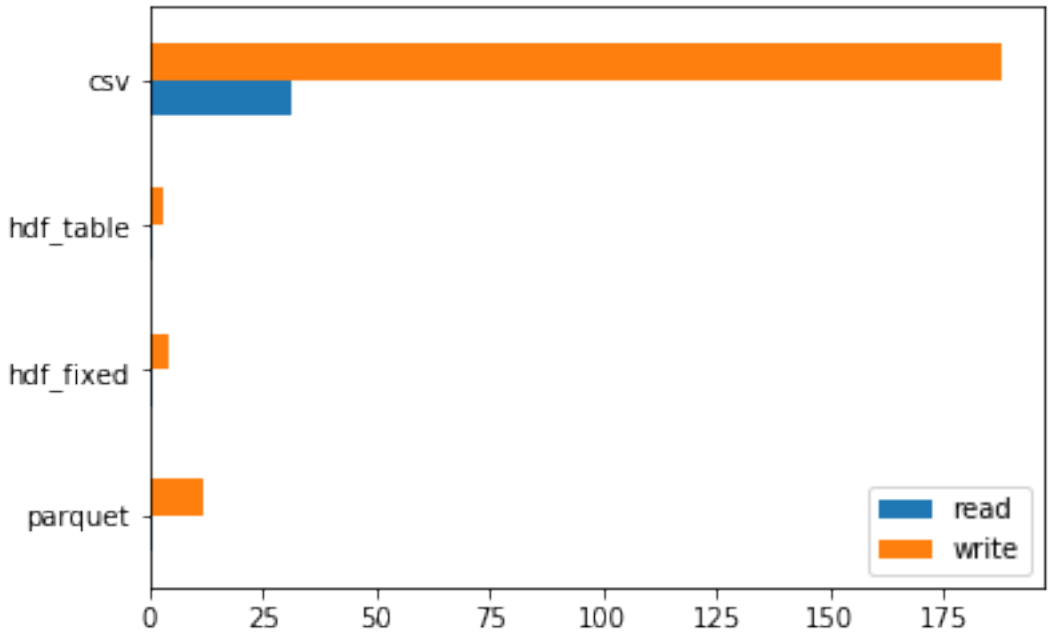
[44]:
```
write = _
```

0.4.4 Results

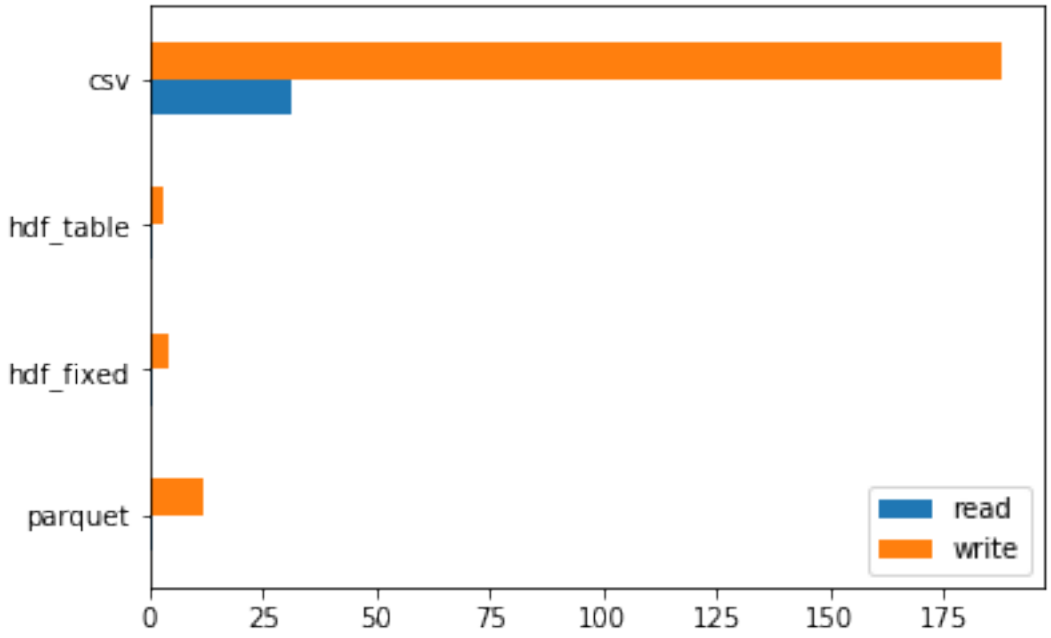
[45]:
```
results['csv'] = {'read': read.all_runs, 'write': write.all_runs, 'size': size}
```

0.5 Store Results

[47]:
```
text_num = pd.concat([pd.DataFrame(data).mean().to_frame(f) for f, data in␣
↪results.items()], axis=1).T
text_num[['read', 'write']].plot.barh();
```



[48]:
```
df = pd.concat([pd.DataFrame(data).mean().to_frame(f) for f, data in results.
↪items()], axis=1).T
# df.to_csv('num_only.csv')
df[['read', 'write']].plot.barh();
```



[ ]:
```
# for f, data in results.items():
#     pd.DataFrame(data).to_csv('{}.csv'.format(f))
```