# Speed it up with … Python?

Martin Kersner

Seoul AI, 2018/03/03

Sofia Heisler Follow
Aug 2, 2017 · 9 min read

# A Beginner's Guide to Optimizing Pandas Code for Speed

If you've done any data analysis in Python, you've probably run across Pandas, a fantastic analytics library written by Wes McKinney. By conferring dataframe analysis functionality to Python, Pandas has effectively put Python on the same footing as some of the more established analysis tools, such as R or SAS.

Unfortunately, early on, Pandas had gotten a nasty reputation for being "slow". It's true that your Pandas code is unlikely to reach the calculation speeds of, say, fully optimized raw C code. However, the good news is that for most applications, well-written Pandas code is *fast enough*; and what Pandas

https://engineering.upside.com/a-beginners-guide-to-optimizing-pandas-code-for-speed-c09ef2c6a4d6

1. Crude looping over DataFrame rows using indices
2. Looping with `iterrows()`
3. Looping with `apply()`
4. ~~Vectorization with Pandas series~~
5. ~~Vectorization with NumPy arrays~~
6. **Looping with `map()`**
7. **Parallel processing**

# Data

```
ImageID,Labels
428e9a49c80b6f23,3671
28e46969241bf0ab,833 3144 3559 3611 3644 3678
d1477bd866d75866,2920 544 2685 3559
00c5baa635e9bd5b,4653
f49a19336a4640ca,4764 549 1256 3364 3678
7c0438950806a174,350 550 701 1836 2776 351 758 2253 474 4349 3671
3a935c3a202697bf,1237 1775 944
d3c01fa2494fc583,2960 1507 3946 4432 1824 1362 1922 1803 2003 1338
```

Dataset consists of image URLs. Many of them are not accessible or download timed out.

https://github.com/openimages/dataset

# Task: Do all images exist?

Create a new column “exist” that holds  boolean value representing existence of image.

```
from pathlib import Path
import pandas as pd

def exists(path: str) -> bool:
  return Path(path).exists()

df = pd.read_csv("annotations.csv")
df["full_path"] = "dataset/" + df.ImageID + ".jpg"  # vectorization with Pandas series
```

# Crude looping over DataFrame rows using indices

```
for idx in range(len(df)):
  df["exist"][idx] = exists(df["full_path"][idx])
  # df.iloc[idx].exist = exists(df.iloc[idx].full_path)
```

1,024 rows, 13.44 seconds

```
for idx in range(len(df)):
  df.loc[idx, "exist"] = exists(df.loc[idx, "full_path"])
```

1,024 rows, 0.64 seconds
5,000 rows, 3.46 seconds

http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

# Looping with `iterrows()`

```
for idx, row in df.iterrows():
  df.loc[idx, "exist"] = exists(df.loc[idx, "full_path"])
```

5,000 rows, 3.76 seconds

# Looping with `apply()`

```
df['exist'] = df['full_path'].apply(lambda path: exists(path))
```

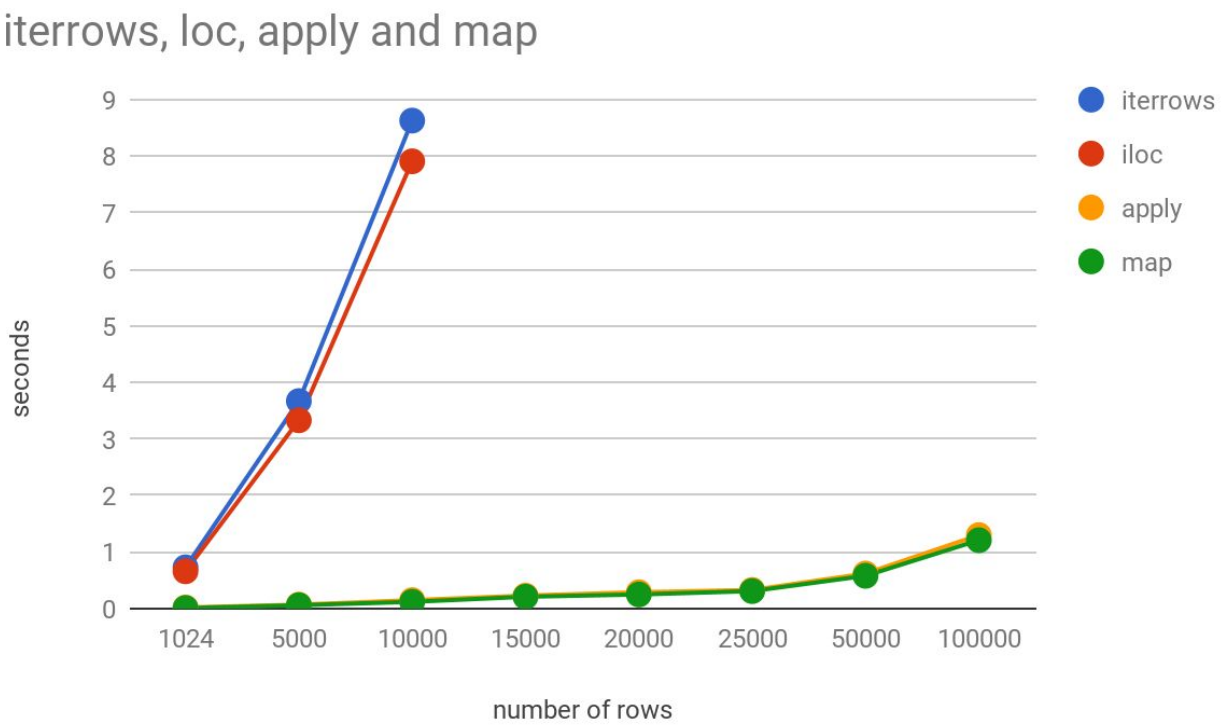
5,000 rows, 0.08 seconds

100,000 rows, 1.31 seconds

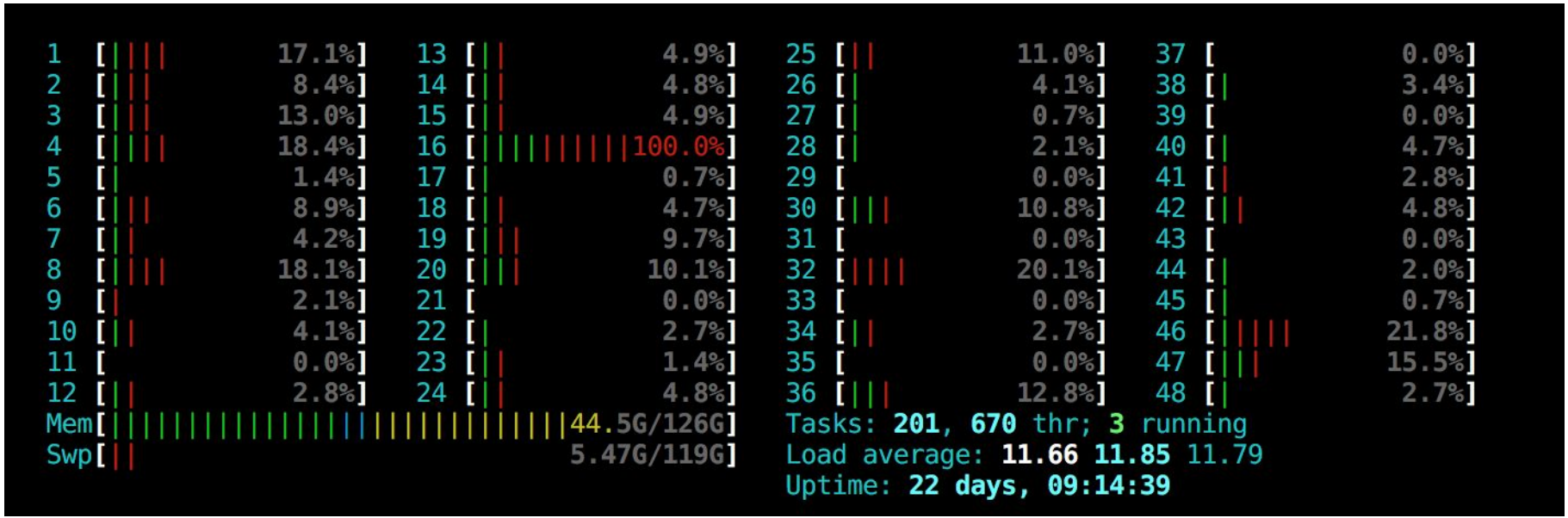
# Looping with `map()`

```
df["exist"] = list(map(exists, df.full_path.tolist()))
```

100,000 rows, 1.25 seconds


iterrows, loc, apply and map
9
8
7
6
5
4
3
2
1
0
seconds
1024
5000
10000
15000
20000
25000
50000
100000
number of rows
iterrows
iloc
apply
map

# Parallel Processing

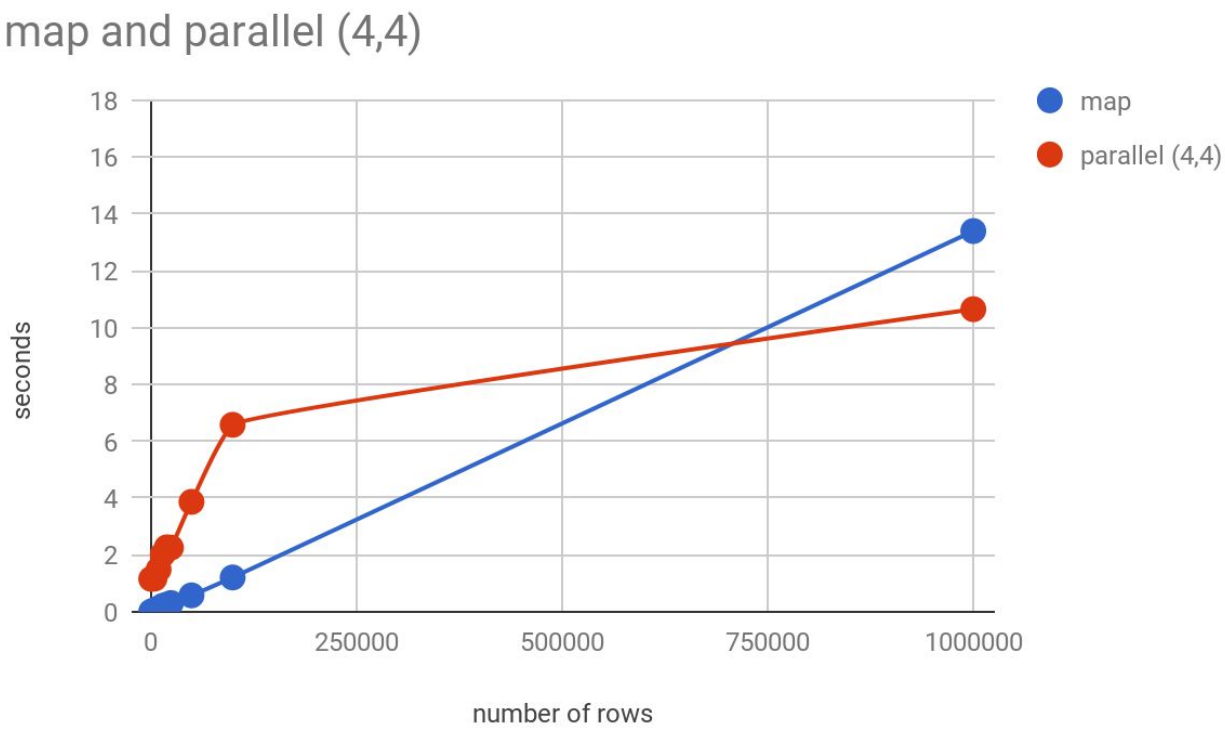


```
def image_exist(data: pd.DataFrame):
  data["exist"] = list(map(exists, list(data.full_path)))
  return data

def parallelize_dataframe(df: pd.DataFrame,
                          fun: Callable,
                          num_partitions: int=4,
                          num_cores: int=4):
  df_split = np.array_split(df, num_partitions)
  with Pool(num_cores) as pool:
    return pd.concat(pool.map(fun, df_split))

df = parallelize_dataframe(df, image_exist)
```

http://www.racketracer.com/2016/07/06/pandas-in-parallel/


map and parallel (4,4)
18
16
14
12
10
8
6
4
2
0
seconds
0
250000
500000
750000
1000000
number of rows
map
parallel (4,4)

# 1,000,000 rows