Remote Linux Cluster Parallelization using Dask in Python

Daniel Huang

12 November, 2021

Python code for parallelized processing and analysis of ~120 gb of zipped Wikipedia Data specifically with regards to Barack Obama during the time period when he won the election the first time from Aug 2008 to Dec 2008.

```
#This is Wiki_Analysis.py:

import dask.multiprocessing
import re
import time
import pandas as pd
import dask.bag as db
if __name__ == '__main__':
    dask.config.set(scheduler='processes', num_workers = 16) # multiprocessing is

    ## This is the full data
    path = '/var/local/s243/wikistats/dated_2017/'
    wiki = db.read_text(path + 'part-00*gz')
    #wiki = db.read_text('part-00*gz')

    def find(line, regex = "Barack_Obama", language = "en"):
        vals = line.split(' ')
        if len(vals) < 6:
            return(False)
        tmp = re.search(regex, vals[3])
        if tmp is None or (language != None and vals[2] != language):
            return(False)
        else:
            return(True)

    obama = wiki.filter(find)

    def make_tuple(line):
        return(tuple(line.split(' ')))

    dtypes = {'date': 'object', 'time': 'object', 'language': 'object',
    'webpage': 'object', 'hits': 'float64', 'size': 'float64'}

    ## Let's create a Dask dataframe.
    ## This will take a while if done on full data.
    df = obama.map(make_tuple).to_dataframe(dtypes)
    type(df)

    ## Now let's actually do the computation, returning a Pandas df
```

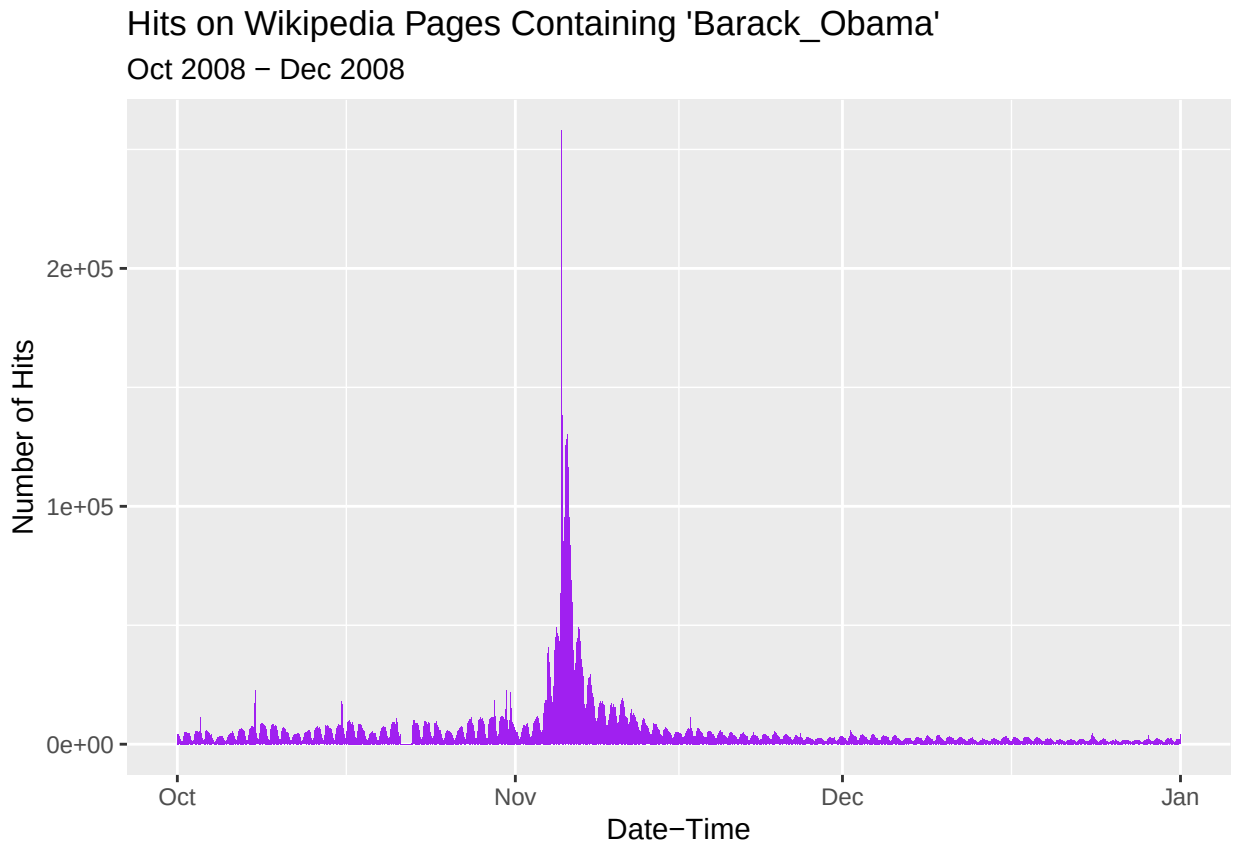
```
    t0 = time.time()
    result = df.compute()
    print(time.time() - t0)
    ## Time is 11364.54 seconds - roughly 3.15 hours

    #Create datetime column, drop unneeded size column
    result.insert(2, 'datetime', result['date'] + ' ' + result['time'])
    result = result.drop(['size'], axis=1)
    result['datetime'] = pd.to_datetime(result['datetime'])

    #Group by date-hours, sum up num of hits
    data = result.groupby(pd.Grouper(key='datetime',freq='H')).sum()

    #Output
    data.to_csv('output.csv')
```

Here is the bash file I used to submit the job using sbatch:

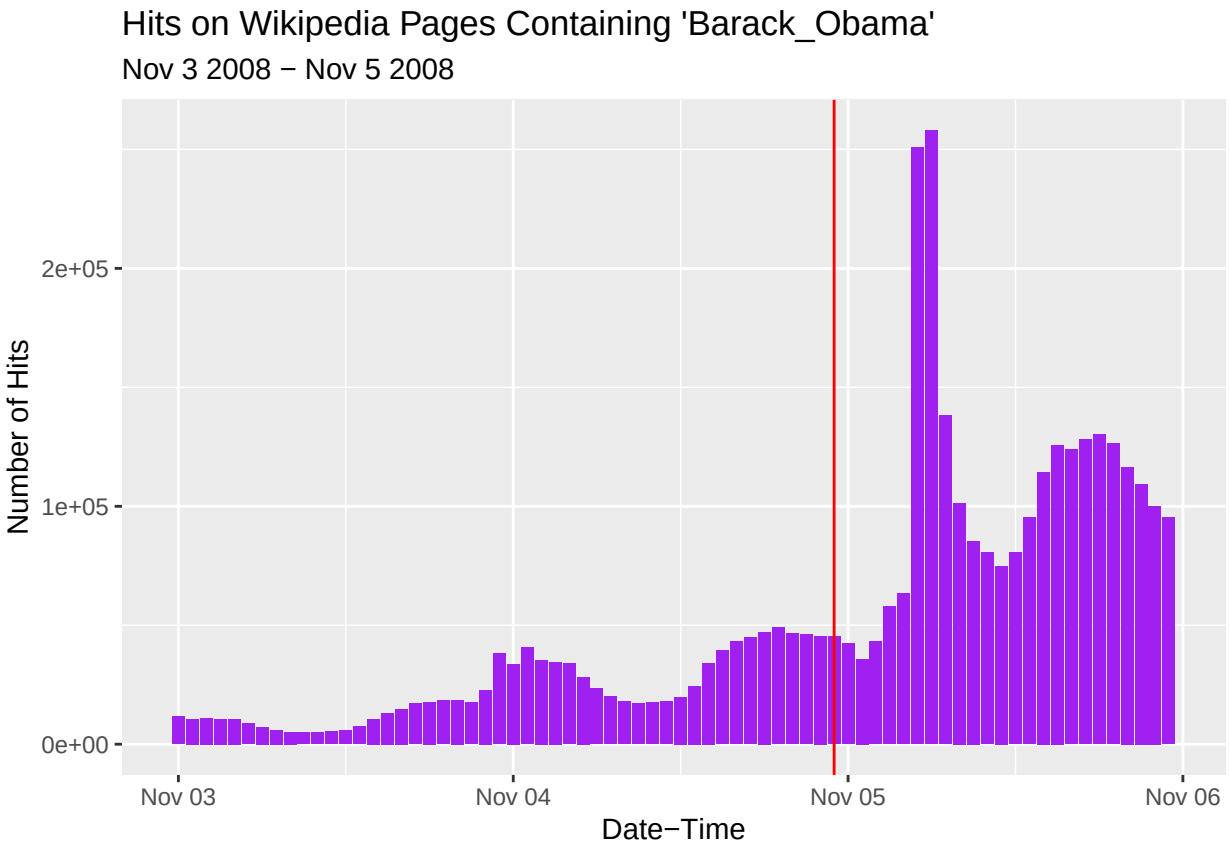
```
#!/bin/bash

####################
# SBATCH OPTIONS
####################
#SBATCH --job-name=DanielWikiSubPy
#SBATCH --partition=low
#SBATCH --error=wiki.err
#SBATCH --output=wiki_analysis_py.out
#SBATCH --cpus-per-task=16

python Wiki_Analysis.py
```

Here is the analysis, once I generated output.csv with all my desired results:

```
obama <- read_csv("output.csv", col_types = "cd")
obama$datetime <- as.POSIXct(obama$datetime,tz="EST")

#Plot all the data
ggplot(data = obama, aes(x = datetime, y = hits)) +
      geom_bar(stat = "identity", fill = "purple") +
      labs(title = "Hits on Wikipedia Pages Containing 'Barack_Obama'",
           subtitle = "Oct 2008 - Dec 2008",
           x = "Date-Time", y = "Number of Hits")
```




```
#Only November 3-5
obama_subset <- filter(obama, datetime >= as.POSIXct("2008-11-03 00:00:00", tz="EST") &
                         datetime <= as.POSIXct("2008-11-05 23:00:00", tz="EST"))

ggplot(data = obama_subset, aes(x = datetime, y = hits)) +
      geom_bar(stat = "identity", fill = "purple") +
      labs(title = "Hits on Wikipedia Pages Containing 'Barack_Obama'",
           subtitle = "Nov 3 2008 - Nov 5 2008",
           x = "Date-Time", y = "Number of Hits") +
  geom_vline(aes(xintercept = as.POSIXct("2008-11-04 23:00:00", tz="EST")), colour = "red", linetype =
```



The solid red line is the time at which Obama's victory was declared, 11 PM Nov 4. We see that there are many more hits after that time. The slight gap is probably because many people went to sleep and decided to check the next day, thus leading to the huge spike the following day.