



Northeastern University

CSYE 7374 - PARALLEL MACHINE LEARNING AND AI SUMMER 2020

HOMework 2

SRISHTI ASHOK MISHRA

001305178

Part 1:

1. Use `Pool.apply()` to get the row wise common items in `list_a` and `list_b`; and print the result. 20 pts `list_a = [[1, 2, 3], [5, 6, 7, 8], [10, 11, 12], [20, 21]]` `list_b = [[2, 3, 4, 5], [6, 9, 10], [11, 12, 13, 14], [21, 24, 25]]`

```
import multiprocessing as mp

list_a = [[1, 2, 3], [5, 6, 7, 8], [10, 11, 12], [20, 21]]
list_b = [[2, 3, 4, 5], [6, 9, 10], [11, 12, 13, 14], [21, 24, 25]]

def common_items(list_1, list_2):
    return list(set(list_1).intersection(list_2))

pool = mp.Pool(mp.cpu_count())
results = [pool.apply(common_items, args=(l1, l2)) for l1, l2 in zip(list_a, list_b)]
pool.close()
print(results[:10])
```

```
[mishra.sr@c0156 homework2]$ ls
example_apply.py  example_starmap.py  pd_sum_of_squares.py  Q1.py  Q2.py  Q3.py  script1.py  script2.py  script3.py
[mishra.sr@c0156 homework2]$ module list
Currently Loaded Modulefiles:
  1) discovery/2019-02-21  2) python/3.8.1
[mishra.sr@c0156 homework2]$ python Q1.py
[[2, 3], [6], [11, 12], [21]]
[mishra.sr@c0156 homework2]$
```

2. Use `Pool.map()` to run the following python scripts in parallel; and print the result. 20 pts Script names: 'script1.py', 'script2.py', 'script3.py' Hint: you can put any content in the three scripts.

```
import os
import multiprocessing as mp

processes = ('script1.py', 'script2.py', 'script3.py')

def run_python(process):
    os.system('python {}'.format(process))

pool = mp.Pool(processes=3)
pool.map(run_python, processes)
```

```
[mishra.sr@c0156 homework2]$ python Q2.py
[1, 2, 3]
Valid Email: srishtimishra101295@gmail.com
Valid Email: mishra.sr@northeastern.edu
Invalid Email: srishtimishra.xyz
Hostname : c0156
IP : 10.99.250.36
End of process 3
End of Process 2
End of process 1
[mishra.sr@c0156 homework2]$
```

3. Normalize each row of 2d array (list) list_c to vary between 0 and 1. Parallelize the function with any subfunction of Pool; and print the result. 20 pts

list_c = [[2, 3, 4, 5], [6, 9, 10, 12], [11, 12, 13, 14], [21, 24, 25, 26]]

```
import multiprocessing as mp

list_c = [[2, 3, 4, 5], [6, 9, 10, 12], [11, 12, 13, 14], [21, 24, 25, 26]]

def normalize_2d_array(list_c):
    min = min(list_c)
    max = max(list_c)
    return [(i - min)/(max-min) for i in list_c]

pool = mp.Pool(mp.cpu_count())
results = [pool.apply(normalize_2d_array, args=(ll, )) for ll in list_c]
pool.close()
print(results[:10])
```

```
[mishra.sr@c0156 homework2]$ python Q3.py
[[0.0, 0.3333333333333333, 0.6666666666666666, 1.0], [0.0, 0.5, 0.6666666666666666, 1.0], [0.0, 0.3333333333333333, 0.6666666666666666, 1.0], [0.0, 0.6, 0.8, 1.0]]
[mishra.sr@c0156 homework2]$
```

Part 2:

A serial code is given here:example_ser.py. Please complete the following tasks:

- Parallelize it with Pool.apply and save as example_apply.py
- Parallelize it with Pool.startmap and save as example_startmap.py

Example_apply.py

```
1 import numpy as np
2 import multiprocessing as mp
3 import time
4
5 # Prepare data
6 np.random.RandomState(100)
7 arr = np.random.randint(0, 10, size=[100000, 1000])
8 data = arr.tolist()
9 #data[:5]
10 start = time.time()
11 def examp01(row, minimum=4, maximum=8):
12     """Returns how many numbers between `maximum` and `minimum` in a given `row`"""
13     count = 0
14     for n in row:
15         if minimum <= n <= maximum:
16             count = count + 1
17     return count
18 # Step 1: Init multiprocessing.Pool()
19 pool = mp.Pool(4)
20 # Step 2: `pool.apply` the `examp01`
21 results = [pool.apply(examp01, args=(row, 4, 8)) for row in data]
22
23 #Step 3 Close Pool
24 pool.close()
25 print('Pool.apply execution:', results[:10])
26 print(f'Execution Time pool.apply(): {time.time() - start} seconds')
27
```

```
[mishra.sr@c0156 homework2]$ ls
example_apply.py example_starmap.py pd_sum_of_squares.py Q1.py Q2.py Q3.py script1.py script2.py script3.py
[mishra.sr@c0156 homework2]$ python example_apply.py
Pool.apply execution: [489, 535, 510, 478, 496, 508, 505, 501, 507, 493]
Execution Time pool.apply(): 31.398431301116943 seconds
[mishra.sr@c0156 homework2]$
```

Example_starmap.py

```
MobaTextEditor
File Edit Search View Format Syntax Special Tools
example_starma... x
1
2 import numpy as np
3 import multiprocessing as mp
4 import time
5 # Prepare data
6 np.random.RandomState(100)
7 arr = np.random.randint(0, 10, size=[100000, 1000])
8 data = arr.tolist()
9 #data[:5]
10 start = time.time()
11 def examp01(row, minimum=4, maximum=8):
12     """Returns how many numbers between `maximum` and `minimum` in a given `row`"""
13     count = 0
14     for n in row:
15         if minimum <= n <= maximum:
16             count = count + 1
17     return count
18
19 pool = mp.Pool(4)
20
21 results = pool.starmap(examp01, [(row, 4, 8) for row in data])
22
23 pool.close()
24 print('Pool.apply execution:', results[:10])
25 print(f'Execution Time pool.starmap(): {time.time() - start} seconds')
26
27
```

```
[mishra.sr@c0156 homework2]$ ls
example_apply.py example_starmap.py pd_sum_of_squares.py Q1.py Q2.py Q3.py script1.py script2.py script3.py
[mishra.sr@c0156 homework2]$ python example_starmap.py
Pool.apply execution: [476, 482, 514, 513, 481, 520, 516, 492, 509, 498]
Execution Time pool.starmap(): 17.904112339019775 seconds
[mishra.sr@c0156 homework2]$
```

Part 3:

A given panda dataframe here pd.py. Please complete the following tasks:

- Define a function to sum the squares of each row, and then calculate the square root, leaving two decimal places for the result.
- Parallelize the function with Pool.imap, and print the output.

```

import numpy as np
import pandas as pd
import multiprocessing as mp

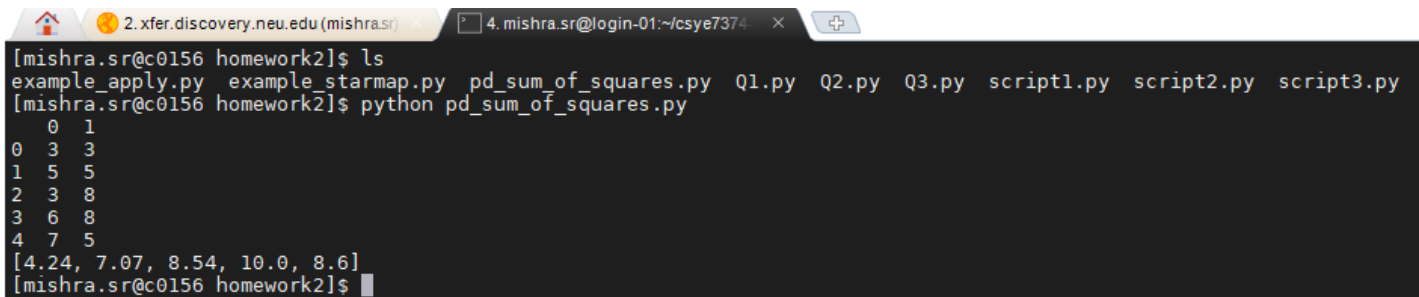
df = pd.DataFrame(np.random.randint(3, 10, size=[5, 2]))
print(df.head())

def sum_of_squares(row):
    return round(row[1]**2 + row[2]**2, 2)**0.5

with mp.Pool(4) as pool:
    result = pool.imap(sum_of_squares, df.itertuples(name=None), chunksize=10)
    output = [round(x, 2) for x in result]

print(output)

```



The terminal window shows the user's current directory and the output of running a Python script. The directory listing shows several files, including 'pd_sum_of_squares.py'. The script output displays a 5x2 DataFrame of random integers, followed by a list of calculated values for the sum of squares function applied to each row.

```

[mishra.sr@c0156 homework2]$ ls
example_apply.py  example_starmap.py  pd_sum_of_squares.py  Q1.py  Q2.py  Q3.py  script1.py  script2.py  script3.py
[mishra.sr@c0156 homework2]$ python pd_sum_of_squares.py
   0  1
0  3  3
1  5  5
2  3  8
3  6  8
4  7  5
[4.24, 7.07, 8.54, 10.0, 8.6]
[mishra.sr@c0156 homework2]$

```