User Manual

Problem Statement:

Consider a monitoring system, which monitors 1000 servers. Each server has 2 CPUs. Each server generates a log for CPU usage every minute. The format is like this:

timestamp	IP	cpu_id	usage
1414689783	192.168.1.10	0	87
1414689783	192.168.1.10	1	90
1414689783	192.168.1.11	1	93

(1) Please write a simulator to generate the logs for one day, say 2014-10-31, just use random numbers between 0% to 100% as CPU usage. The generator should write data to files in a directory. The timestamp is Unix time.

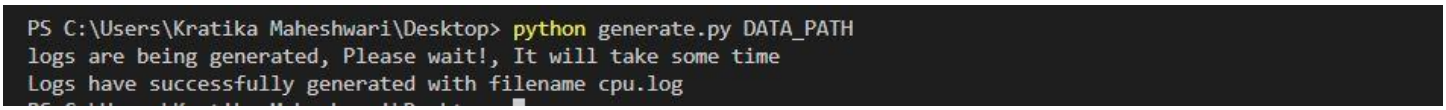
(2) Please create a command line tool which takes a directory of data files as a parameter and lets you query CPU usage for a specific CPU in a given time period. It is an interactive command line tool which read a user's commands from stdin. The tool should support two commands. One command will print results to stdout. Its syntax is QUERY IP cpu_id time_start time_end. Time_start and time_end should be specified in the format YYYY-MM-DD HH:MM where YYYY is a four digit year, MM is a two digit month (i.e., 01 to 12), DD is the day of the month (i.e., 01 to 31), HH is the hour of the day, and MM is the minute of an hour. The second command to support is EXIT. It will exit the tool.
The tool may take several minutes to initialize, but the query result should be returned within 1 second.

Solution:

I have executed the following tasks using python. There are 2 python files generate.py and query.py for logging the data and fetching them, respectively. It took me approximately 5 hours to complete and provide the optimized solution for the given problem statement.
To run the model, first run generate.py using command **python generate.py DATA_PATH,** the log file named cpu.log will be generated inside the directory DATA_PATH which will be created in the current directory where other python files are present , after that run query.py file using command **python query.py DATA_PATH** to fetch the query records.

(1) To generate the logs for one day, execute the 'generate.py' file and logs get generated in the directory DATA_PATH (which will be created inside present directory) with filename 'cpu.log'.

Screenshot 1:



Screenshot 2:

```
Timestamp IPAddress CPU_Id CPU_Usage
1592779430 192.0.0.0 0 8
1592779430 192.0.0.0 1 23
1592779430 192.0.0.1 0 2
1592779430 192.0.0.1 1 30
1592779430 192.0.0.2 0 56
1592779430 192.0.0.2 1 56
1592779430 192.0.0.3 0 87
1592779430 192.0.0.3 1 88
1592779430 192.0.0.4 0 25
1592779430 192.0.0.4 1 98
1592779430 192.0.0.5 0 15
1592779430 192.0.0.5 1 11
1592779430 192.0.0.6 0 53
1592779430 192.0.0.6 1 43
1592779430 192.0.0.7 0 25
1592779430 192.0.0.7 1 26
1592779430 192.0.0.8 0 62
1592779430 192.0.0.8 1 41
1592779430 192.0.0.9 0 77
1592779430 192.0.0.9 1 46
```

A total of 2,880,000 records get generated in cpu.log file.

(2) To query CPU usage for a specific CPU in a given time period, run file 'query.py'. It would ask for the type of request the user want to make between 'QUERY' with the syntax QUERY IP cpu_id time_start time_end and 'EXIT'. Please find below screenshots of both expected and unexpected input arguments.

EXPECTED CASES:

Screenshot 3: When the user requests logs for QUERY 192.0.0.57 1 2020-06-21 23:00 2020-06-22 06:30

When the user requests logs for QUERY 192.0.1.23 0 2020-06-21 23:59 2020-06-22 00:15

```
PS C:\Users\Kratika Maheshwari\Desktop> python query.py DATA_PATH

Enter your request:
QUERY 192.0.1.23 0 2020-06-21 23:59 2020-06-22 00:15
CPU0 usage on 192.0.1.23:
[('2020-06-21 23:59', '7%'), ('2020-06-22 00:00', '66%'), ('2020-06-22 00:01', '0%'), ('2020-06-22 00:02', '15%'), ('2020-06-22 00:03', '14%'), ('2020-06-22 00:04', '0%'), ('2020-06-22 00:05', '3%'),
 ('2020-06-22 00:06', '92%'), ('2020-06-22 00:07', '0%'), ('2020-06-22 00:08', '39%'), ('2020-06-22 00:09', '13%'), ('2020-06-22 00:10', '74%'), ('2020-06-22 00:11', '76%'), ('2020-06-22 00:12', '83%
'), ('2020-06-22 00:13', '40%'), ('2020-06-22 00:14', '40%')]
Total time taken by query:0.030913591384887695 seconds

Enter your request:
exit
PS C:\Users\Kratika Maheshwari\Desktop>
```

In the above screenshots the **total time taken** to fetch the logs with time stamps ranging from 23:00 to 06:30 is **0.3 seconds approx** (highlighted in red color) and for second query ranging from 23:59 to 00:15 is **0.03 seconds approx**

Screenshot 4: When the user requests to exit.

```
Enter your request:
EXIT
PS C:\Users\Kratika Maheshwari\Desktop> []
```

<u>UNEXPECTED CASES</u>:

Screenshot 5: When the IP Address is out of range.

```
PS C:\Users\Kratika Maheshwari\Desktop> python query.py DATA_PATH
Enter your request:
QUERY 192.10.1.23 0 2020-06-21 23:59 2020-06-22 00:15
IP Address out of range. Sould be between 192.0.0.0 to 192.0.3.231
```

Screenshot 6: When the CPU Id is not either 0 or 1.

```
Enter your request:
QUERY 192.0.0.57 2 2020-06-21 23:45 2020-06-22 00:15
Invalid CPU ID, Should be either 0 or 1 !!

Enter your request:
QUERY 192.0.0.57 2 2020-06-19 23:45 2020-06-20 00:15
Invalid CPU ID, Should be either 0 or 1 !!
```

Screenshot 7: When the dates are not present in the logs generated.

```
Enter your request:
QUERY 192.0.0.57 1 2020-06-19 23:45 2020-06-20 00:15
Request is out of log data
```

Screenshot 8: When the start time should be earlier than the end time.

```
Enter your request:
QUERY 192.0.0.57 1 2020-06-20 23:45 2020-06-20 00:15
Start date time should be earlier than End date time
```

Screenshot 9: When the user inputs wrong number of arguments.

```
Enter your request:
QUERY 192.0.0.57 1 2020-06-19 23:45 2020-06-20
Invalid number of arguments, please check again!
```

Screenshot 10: When the user request type is neither ‘Query’ nor ‘Exit’.

```
Enter your request:
abcd
Invalid request type, Should be either QUERY format or EXIT !!
```

Screenshot 11: When input start and end date are of future and not present in log data.

```
PS C:\Users\Kratika Maheshwari\Desktop> python query.py DATA_PATH

Enter your request:
QUERY 192.0.1.23 0 2020-06-23 23:59 2020-06-24 00:15
Logs are earlier than 2020-06-22 23:36 , please change your start date
```