ANALYSIS REPORT

BENCHMARK SERVER

Version:	**1.3**
Status:	Approved
Approver:	Ngô Thái Bình Nguyễn Thị Diễm Trang
Author:	Nguyễn Bảo Nguyên Quách Hoàng Minh Nguyễn Vũ Anh Thư

netcompany

Document history

Version	Date	Author	Status	Remarks
1.0	24-02-2022	Nguyễn Bảo Nguyên	Draft	
1.1	02-05-2022	Nguyễn Vũ Anh Thư Quách Hoàng Minh	Draft	Edit
1.2	12-05-2022	Nguyễn Bảo Nguyên	Draft	Reviewed and add a table to compare results between experiments.

References

Reference	Title	Author	Version

Table of contents

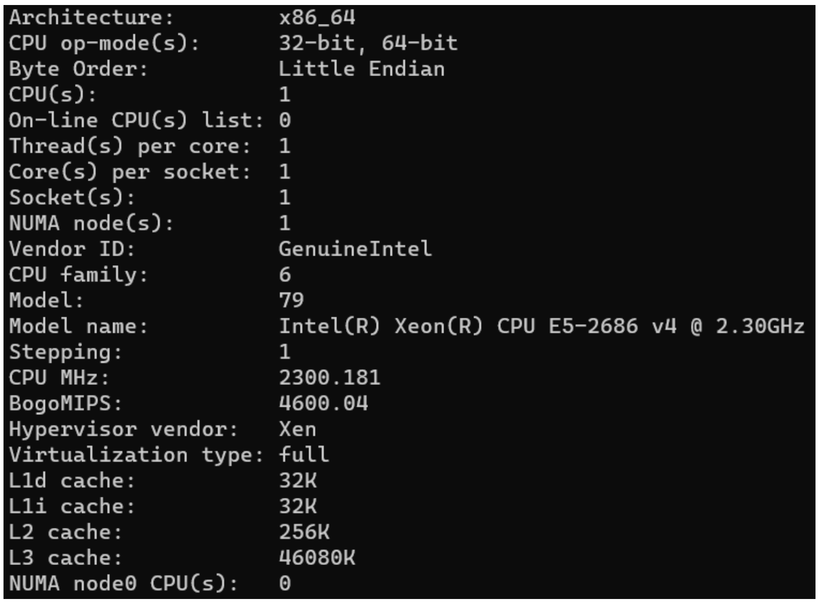
1. Problems

The primary objective of this report is to examine the server's performance in predicting stock prices. When the script to forecast the price of 300 shares was run consecutively on the server, the CPU got stuck, leading the server to crash.

We executed some experiments to collect statistics regarding server performance, and analysed them to determine, from there, we can make the decision to set up a cronjob's configuration to run scripts appropriately.

2. Experiments and Results

The following is some information on the CPU on the Amazon EC2 server we were provided:



```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list: 0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
Stepping:              1
CPU MHz:               2300.181
BogoMIPS:              4600.04
Hypervisor vendor:     Xen
Virtualization type: full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              46080K
NUMA node0 CPU(s):     0
```

Figure 1: Server Information

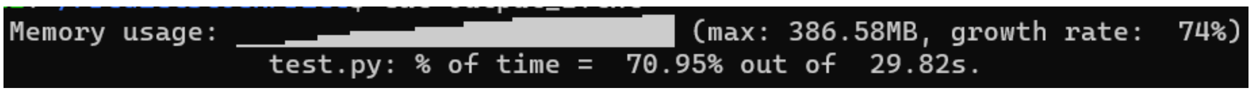
We can't implement multithreading based on the server's information, hence we decided to evaluate with the following scenarios:

- One stock at a time
- Sequential two stocks
- Sequential two stocks with interval and release resources
  - Interval = 5 seconds
  - Interval = 15 seconds

2.1. One stock at a time

Firstly, I execute the script to predict the price of a single stock at a time. From there I can evaluate how many seconds a stock runs and how many resources are used.

- Peak memory usage is 386.58MB
- Peak CPU utilization is 92.9 %
- Execution time is 29.82s



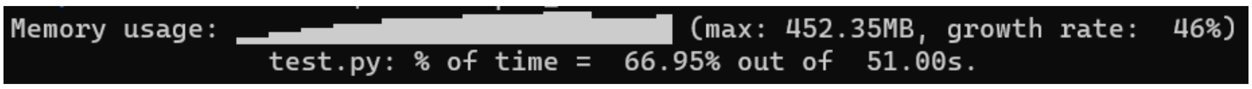
```
Memory usage:                          (max: 386.58MB, growth rate:  74%)
   test.py: % of time =  70.95% out of  29.82s.
```

Figure 2: Memory Usage when run one stock at a time

2.2. Sequential two stocks

Since the server has only 1 core and 1 thread, we can't do multi-threaded testing. Therefore, we decided to run the predictions for the stocks sequentially. In the second test, I made 2 stock price predictions sequentially.

- Peak memory usage is 460.82MB
- Peak CPU utilization is 100.0 %
- Execution time is 51.0s



```
Memory usage:                          (max: 452.35MB, growth rate:  46%)
   test.py: % of time =  66.95% out of  51.00s.
```

Figure 3: Memory usage when run sequentially 2 stocks

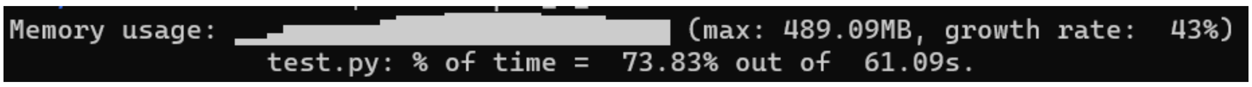
2.3. Sequential two stocks with interval and release resources

While doing the previous test, we noticed that there is a need for a break between stocks to avoid CPU spikes. In addition, releasing resources is a necessity to free up memory.

a. Interval = 5 seconds

With a delay between the two of five seconds, we made two successive stock forecasts and collected the results below.

- Peak memory usage is 490.76MB
- Peak CPU utilization is 93.9 %
- Execution time is 61.09s



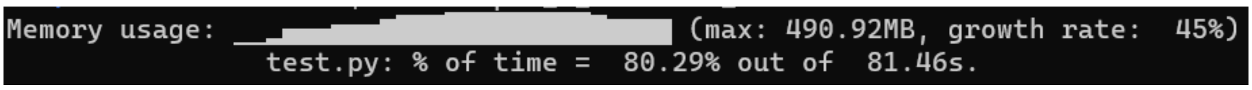
```
Memory usage:                          (max: 489.09MB, growth rate:  43%)
   test.py: % of time =  73.83% out of  61.09s.
```

Figure 4: Memory usage when run sequentially 2 stocks with interval = 5s and release resources

b. Interval = 15 seconds

To check if the delay time between stocks affects CPU performance, I increase the interval to 15s.

- Peak memory usage is 490.92MB
- Peak CPU utilization is 98.9 %
- Execution time is 81.46s



```
Memory usage:                          (max: 490.92MB, growth rate:  45%)
   test.py: % of time =  80.29% out of  81.46s.
```

Figure 5: Memory usage when run sequentially 2 stocks with interval = 15s and release resources

We have run those experiments sometimes and obtained the average values in the table below:

	One stock at a time	Sequential 300 stocks	Sequential two stocks	Sequential two stocks with interval and release resources	
				Interval = 5	Interval = 15
Peak memory usage	386.58MB	Server crashed while executing	460.82MB	490.76MB	490.92MB
Peak CPU utilization	92.9%	100%	100.0%	93.9%	98.9 %
Execution time	29.82s	Server crashed while executing	51.0s	61.09s	81.46s

3. Conclusion

- Based on the results of the previous attempts, we decided to run stocks sequentially, releasing resources after completing one stock and sleeping for 5 seconds after running the next.
- We tried to run 300 stocks continuously, however as we reached roughly 83 stocks, the server crashed. To avoid CPU overloading, we divided the 300-stock list into six smaller lists, each with 50 stocks.
- And since each stock costs between 35 and 50 seconds (including sleeping 5s), a total of 50 stocks will cost around 2500 seconds.
- In conclusion, we depend on the results to set up a cronjob to execute the script automatically, with each list (50 stocks) running 1 hour 30 minutes apart to avoid CPU overloading.