15/12/2017

CA Write-Up

Inspecting the results from the bash script through four calculated graphs

Martin Casey
X00119025
ITT DUBLIN

Contents

Overview

The following document details the results obtained from a bash script, that was designed to perform a load test that would be killed after 10 seconds once a an mpstat command was completed.

```
CO N Idle
47 1 85.69
82 2 76.28
117 3 73.64
```

The mpstat command would generate a report, containing the virtual machine details that are used to calculate the formulas discussed in this document. The details are the values for “CO”, “N” and “Idle”.

System details

This script was tested on a fedora 64-bit operating system, running in a virtual machine in oracle virtual box, running on top of windows 10. The windows 10 PC which is running the virtual machine contains 8 CPU processors. The virtual machine used for this test, utilities one CPU at 100% of the execution cap.

The Value of N

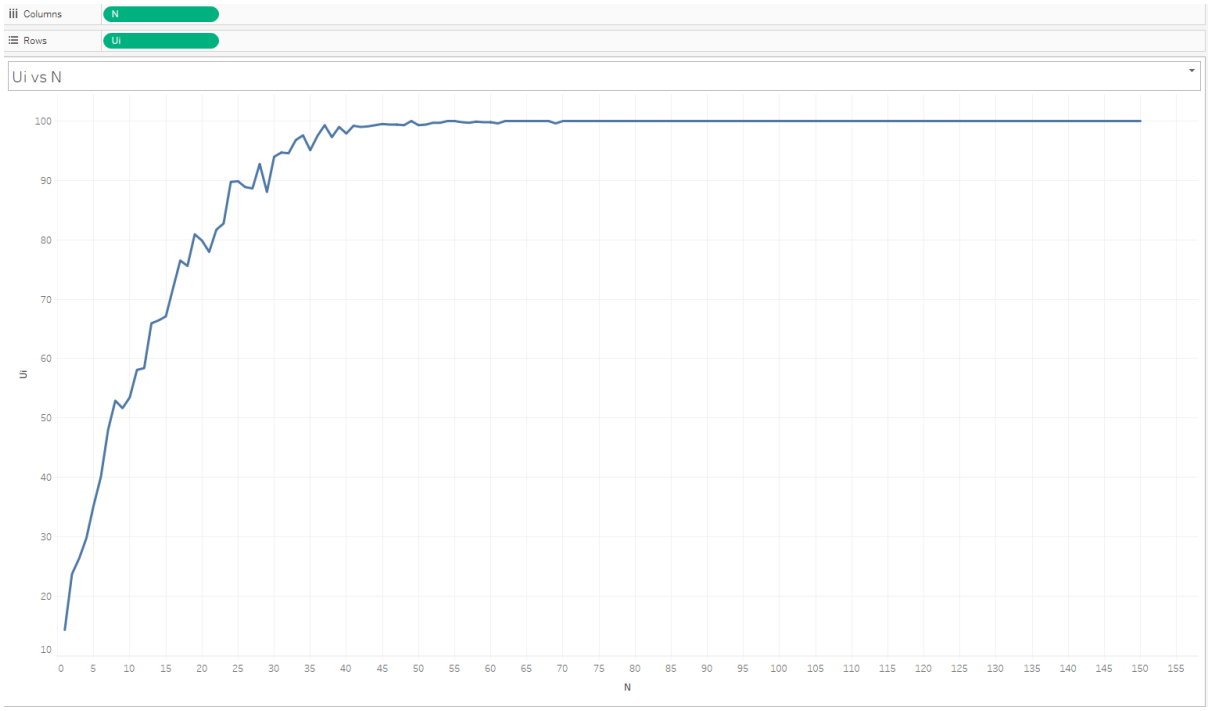
In the following outlines, the value of N represents the number of users in the bash script which performed a load test. The number of users was determined by an incrementing value in a for loop for the bash script, which runs until it has surpassed a maximum of 150 users in the script.

```
#Linux for loop
for i in {1..150};
do
```

The Four Graphs

Ui Vs N

Ui Formula: Ui = 100 – Idle;



The above graph represents the trend that has occurred in the utilization of the CPU resource in the virtual machine during the execution of the bash script. The trend from the graph shows that at the start of the bash script, as the number of users utilizing the CPU increased, so did the amount of resources used by the CPU. At around the point of eight users, the CPU resource utilization surpasses 50%. At this point the virtual machine would start to experience some slight slowdown, when doing other actions, but still have some resources left for a user to perform those actions.

After the eight user, the CPU utilization continues to rise, but now experiences occasional decreases in resources used. This trend in rises and slight falls in resource use continues up to 41 users, where CPU resource utilization has reached 99.20% for the virtual machine. After this, the resource usage starts to even out, now that there are very few resources remaining for the system to run separate tasks, as the bash script runs.

The system reaches 100% utilization at 49 users. This means that all of the systems resources in the virtual machine are devoted to completing the bash script at this point in time. At 50 users on onward the resource usage lessens slightly as more users are added, but utilization never falls below 99.60%.

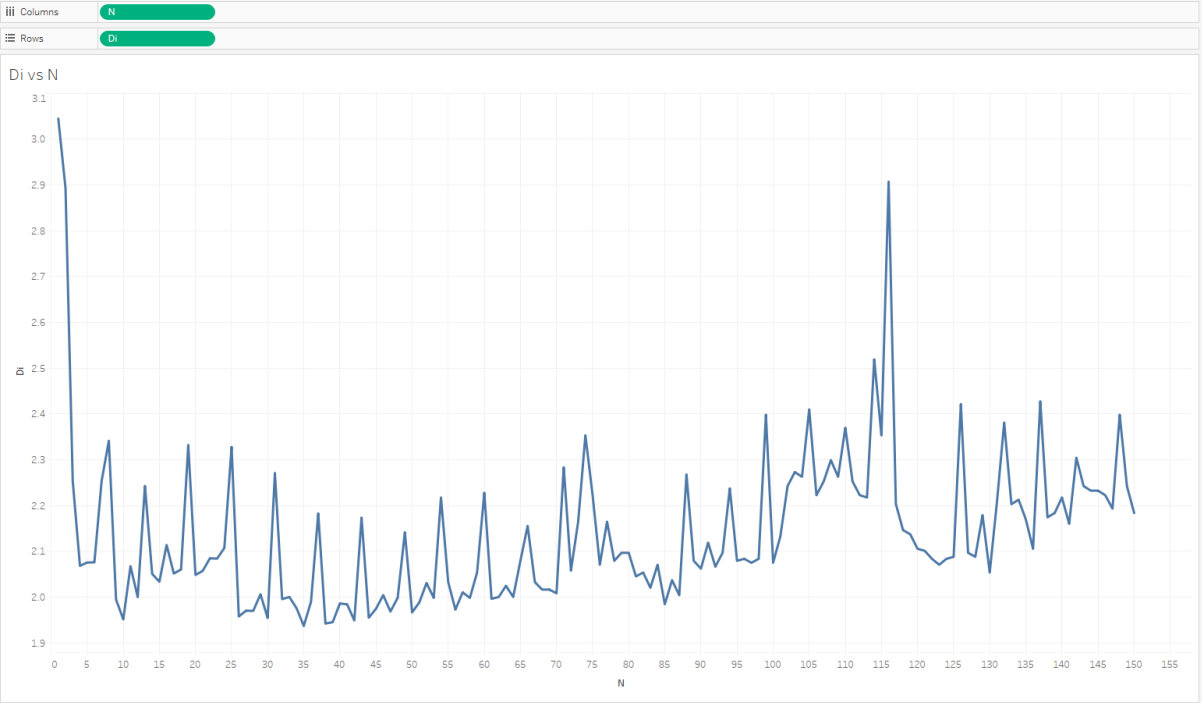
The author believes that less resources were used by the bash script around this point, due to the author trying to regain access to the virtual machine, after they were logged out of it, due to a prolonged period of inactivity while they waited for the script to finish.

Finally, once the system reaches 70 users, resource allocation returns to 100%, and stays that way until the bash script runs through all 150 users. At this stage, all the CPUs resources are being devoted to just running this bash script, so no other task are being performed at this time, until all users are finished.

Di vs N

Di Formula: Di = Ui * T / CO;

T in this context has the value of 10 seconds.



The above graph details the trend that has occurred for the service demand of the system while the system is running the bash script. For around the start until the script reaches four users, the decline in service demand is gradual and steady. At five users the service demand of the system begins to experience steep rises, followed by steep falls. This is likely due to the system completing and then requesting new load tests as the bash script increases the number of users. The first significant drop in service demand while the script is underway is experienced when the system reaches 26 users.

The system then continues on a continual path of rises and falls as more tasks are completed and ended, and more users are added to the bash script. At around 103 users the system

begins to enter a stage in where the service demand remains high, up until the largest rise in service demand on the system of 2.907% with 116 users.

The author believes this large rise is due to the virtual machine system likely completing multiple tasks all at once. Such as re-granting access to the author after they were logged out, or registering mouse movement while the author was checking on the system.

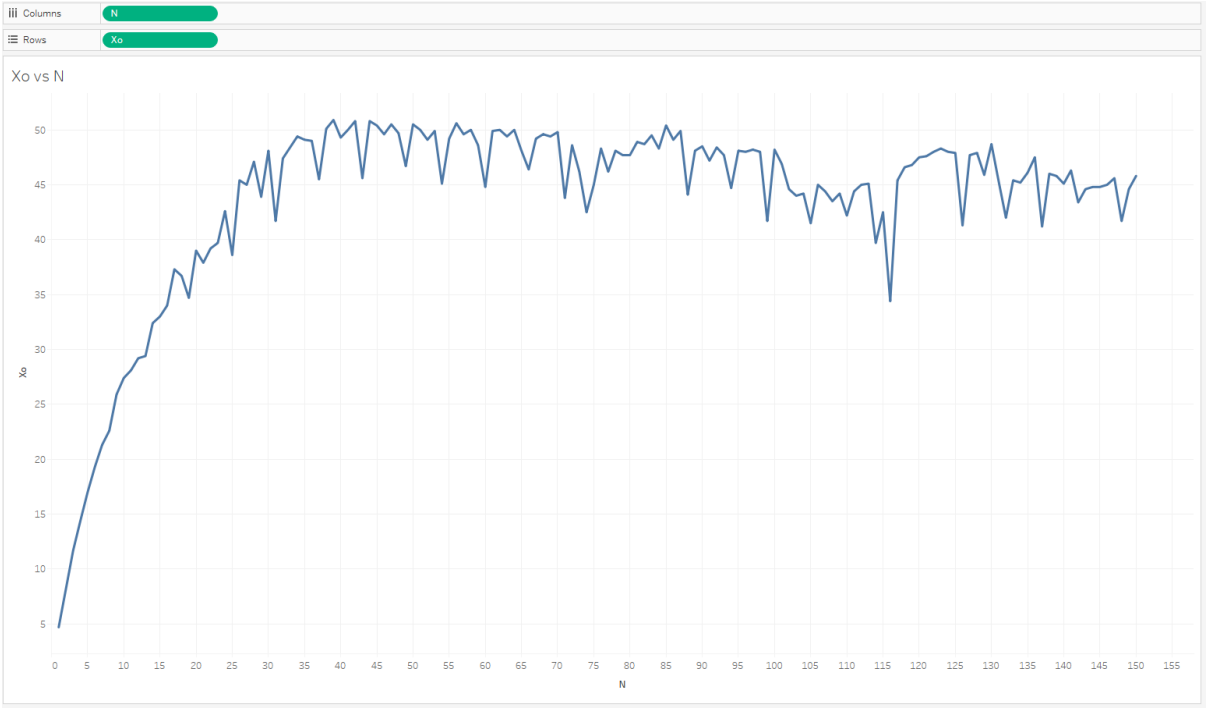
```
./runtest.sh: line 6:  4452 Terminated              ./loadtest $i
./runtest.sh: line 6:  4609 Terminated              ./loadtest $i
./runtest.sh: line 6:  4656 Terminated              ./loadtest $i
```

After the rise comes a very steep decline followed by a large gap between the next rise in service demand. At this point in time the system was likely very not requesting as many resources, and was more concerned in completing tasks that were underway, than creating new tasks. After the rise in service demand that comes after the steep gap, the system continues back on a similar rise and fall usage of service demand, as the bash script reaches 150 users.

Xo vs N

Xo Formula: Xo = Co / T;

T in this context has the value of 10 seconds.



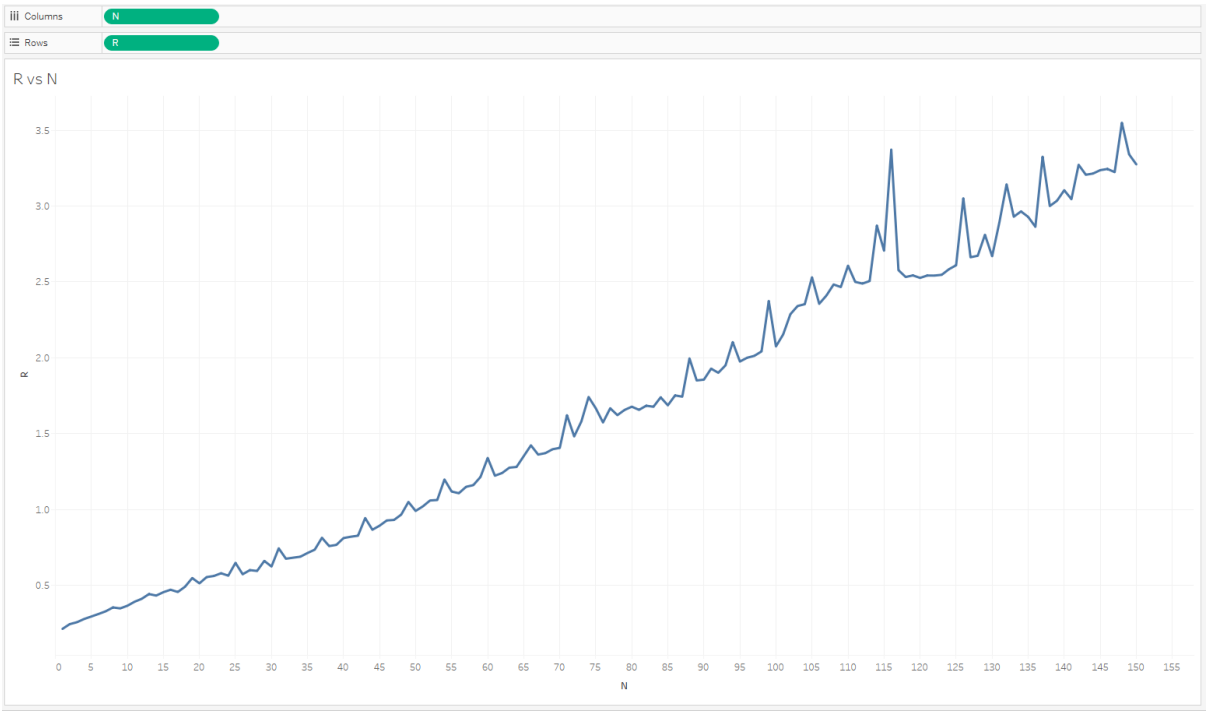
The above graph details the trend for system throughput that has taken place, during the execution of the bash script in the virtual machine. The amount of data being processed by the system seems to rise consistently in a gradual rise, until the system surpasses 17 users. Afterwards it experiences a slight drop, before rising again, only this time the rise is not gradual, and is followed by occassional drops.

Once the system reaches 30 users, the rises and drops in throughput begin to boht gain in size. It is around this period were the system reaches its maximum throughput of 50.90% at 39 users. This trend continues until the system reaches 70 users. Around this time the distance between the significant rises, and deep drops widens, leading to the steepest drop of system throughput to 34.40% at 116 user. The author believes this steep drop in througput may be due to the system reaching a stage in where the only task it was actively processing was the bash script, which was performing loadtests and generating reports. Interrestingly this steep fall in throughput mirrors the steep rise in service demand shown in the previous graph, with both even at the same number of users (116).

After the rise in throughput that follows on from the stepest drop, the gaps between the rises and falls in throughput begin to decreass again, as the system throughput slowly decreases as the system finishes off with the final users.

R vs N

R Formula: N / Xo;



The above graph represents the trend that has occurred in the system for response time when running the bash script in the virtual machine. Generally as the number of users increases, the response time seems to increase with it. This is likely due to the fact that the bash script needs to run more load tests as the number of users increases.

```
#Value i is the number of users on the system
./loadtest $i &
```

The rise in response time is relatively smooth and gradual. It is at around 25 users in the bash script in where the response time begins to start experiencing occasional small spikes in rises for response time.

It is at around 74 users where the system starts to experience bigger rises in response time, in this case it being a response time of 1.741 seconds. The system then continues with a gradual rise and decrease in response time with bigger and bigger spikes. The system reaches its longest response time of 3.372 seconds at 116 users. This spike in response time mirrors the spikes for both the throughput and service demand experienced in the previous graphs. This evidence can there for be used to confirm that the virtual machine system experienced noticeable downturn in performance once the bash script it was running reached 116 users.

After its steepest rise in response time, the system then experiences its steepest decline also. This decline is followed by a period of stagnation, in where the response time barely rises or falls. This period of stagnation mirrors the large gap for system throughput detailed in the prior graph, and the large gap in service demand in the second graph of the document. This can be used to infer that the overall virtual machine system after experiencing it's largest rises in both service demand, response time, and largest decline in throughput; it then goes on to experience a period of stagnation, in where it is primarily concerned with completing a set of task it currently has underway.

After this period of stagnation the system re-experiences steep rises and shorter falls in response time. This seems to show that the system was taking longer to finish some tasks as the number of users increased, which affected the amount of resources the system could work with, as detailed in the first graph. After this the system reaches 150 users and the bash script finishes.

Conclusion

From running the bash script up to 150 users, the general trend that can be found in the system is that, as more tasks are initiated as the number of users increases, the performance of the overall virtual machine will decline as the script runs to completion. The exceptions to this as detailed in the second, third and fourth graphs, can in most likely hood be attributed to the system completing multiple tasks at once after a significant delay in task completion, due to a lack of resources for the CPU.

Appendix

Bash Script (runtest.sh)

```
#!/bin/bash

#Writes the headings of the 3 values to the file datafile.dat
echo "CO" "N" "Idle" > results.dat
#Linux for loop
for i in {1..150};
do
#Run load test in background
#loadtest runs forever
#Value i is the number of users on the system
./loadtest $i &
#Collect and output CPU Utilization
#mpstat command runs for 10 seconds each, and generates 1 report every 10
seconds
idle=`mpstat 10 1 -o JSON | jq '.sysstat.hosts[0].statistics[0]."cpu-
load"[0].idle'`
#Read from the file synthetic.dat to get the CO
#The < ensures wc -l does not also print the name of the file
CO=`wc -l < ./synthetic.dat`
#Append the values to the file datafile.dat
        echo $CO $i $idle >> results.dat
#Kills loadtest
pkill loadtest
done
```

Reference

- **StackOverflow (2017). How to get “wc -l” to print just the number of lines without file name? [online]. Available at: https://stackoverflow.com/questions/10238363/how-to-get-wc-l-to-print-just-the-number-of-lines-without-file-name [Accessed 15th December 2017]**