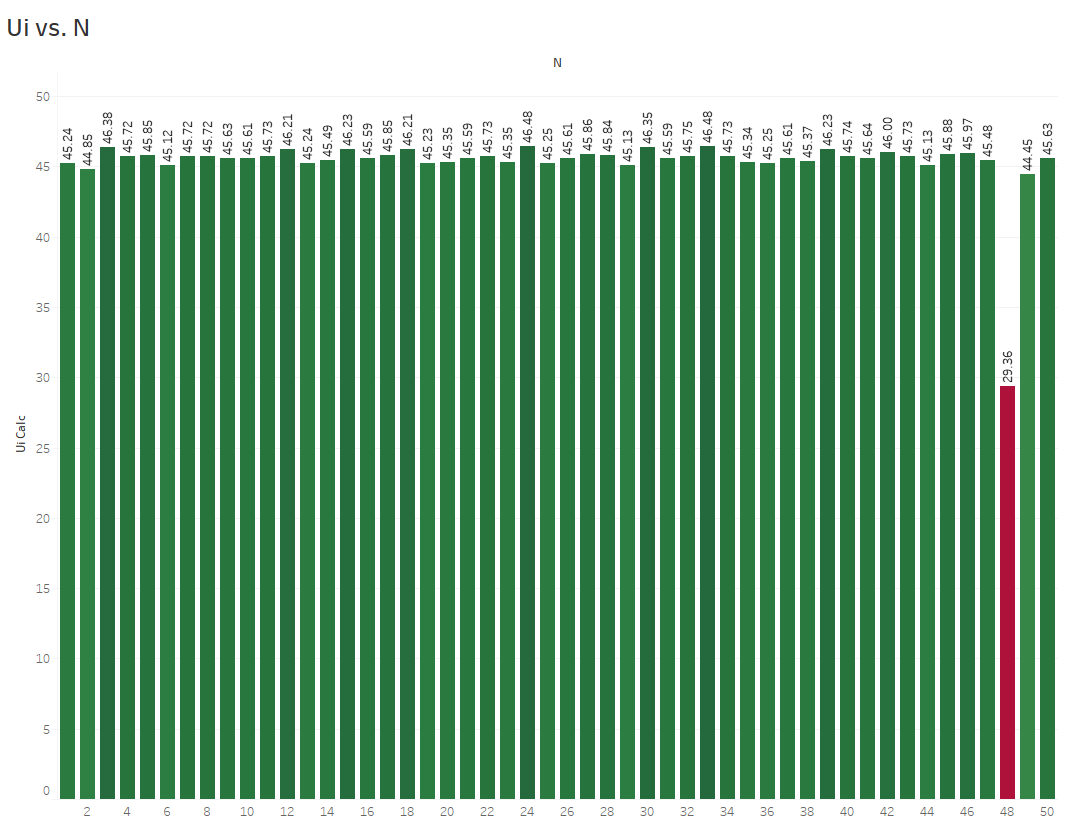
**EPA CA2 – X00129654 – Report**

**Plot Discussion**

After gathering data for the load test bash script and loading the data into Tableau, I created 4 graphs to display my results. The first graph shows the utilization (Ui vs. N) and this was calculated using the following formula:



This graph illustrates that the utilization of the CPU generally ranged between 44 – 46 for each iteration of the loadtest, except for one outlier on the 48th iteration where the Ui vs. N value was only 29.36, as seen below.



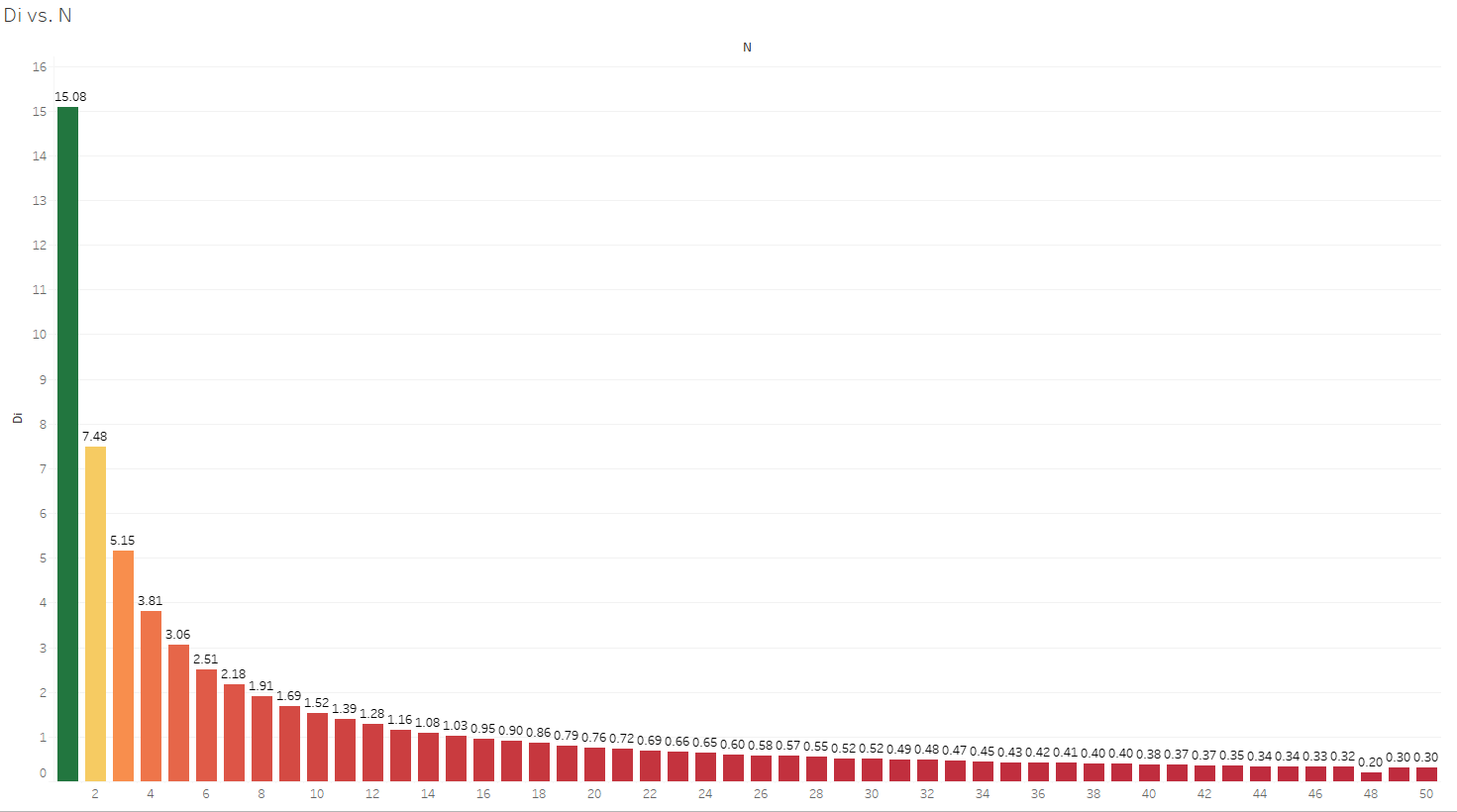
In my opinion, this was due to a temporary lag within the VM while the script was running causing CPU to plummet momentarily.

The second visualization I created was centered around working out the Service Demand

(Di vs. N) and I used the following calculation: Di = Vi x Si = Ui / X0



This showed a large drop-off of the value of Service Demand after the first few iterations, eventually levelling out at between 0.5 and 1.5 as seen below.

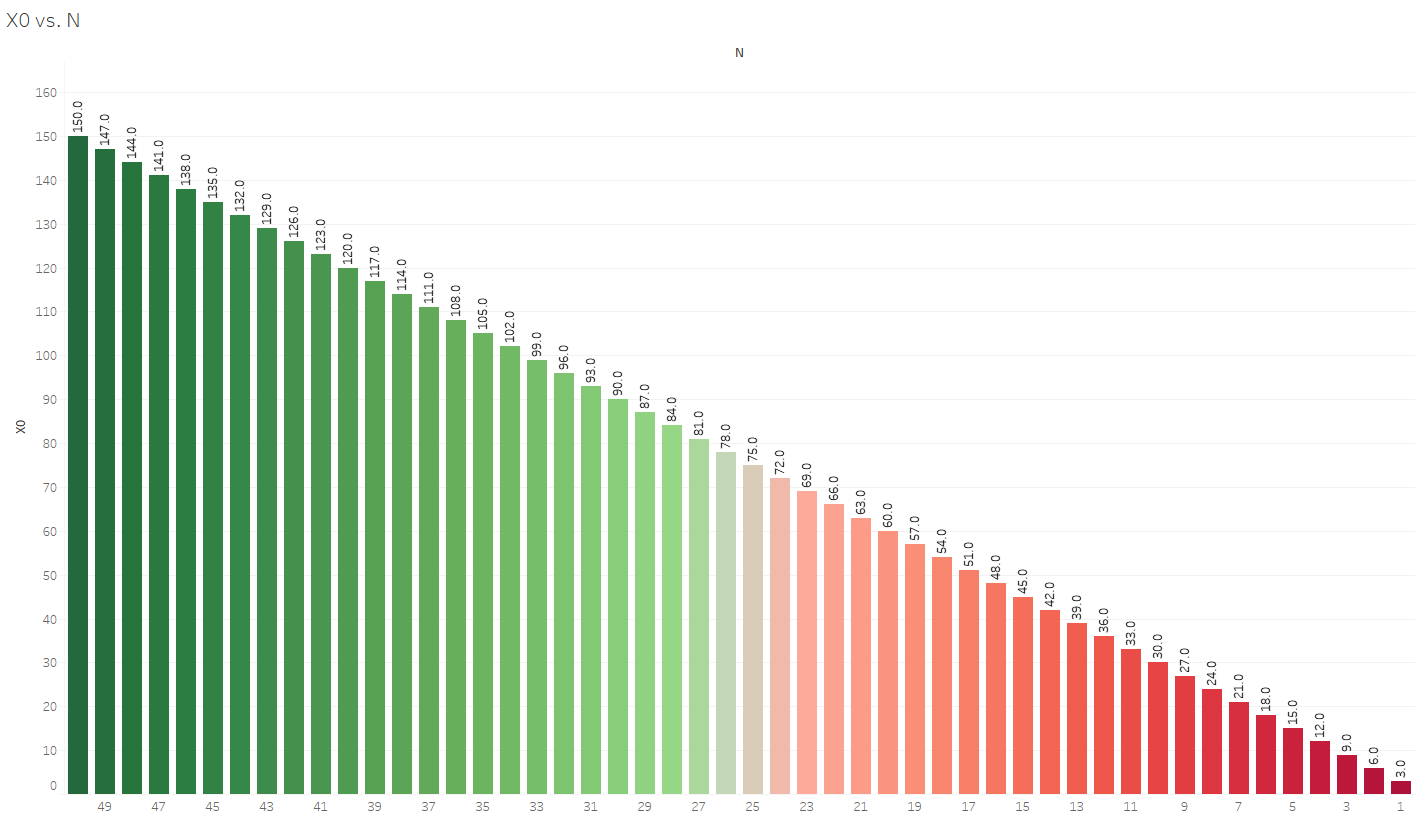


This was caused by the fairly rapid increasing of the c0 value in the script coupled with the consistent idle values in each iteration causing the Service Demand to eventually plateau when the disparity between values had reached such a point. This becomes very apparent after the 10th iteration of the load test.

Moving on, the third graph I created is focused calculating system throughput (X0 vs.N) and was worked out using the following formula: X0 = C0 / T



This graph shows a clear trend as the value of system throughput continuously increases as each iteration finishes as seen below.

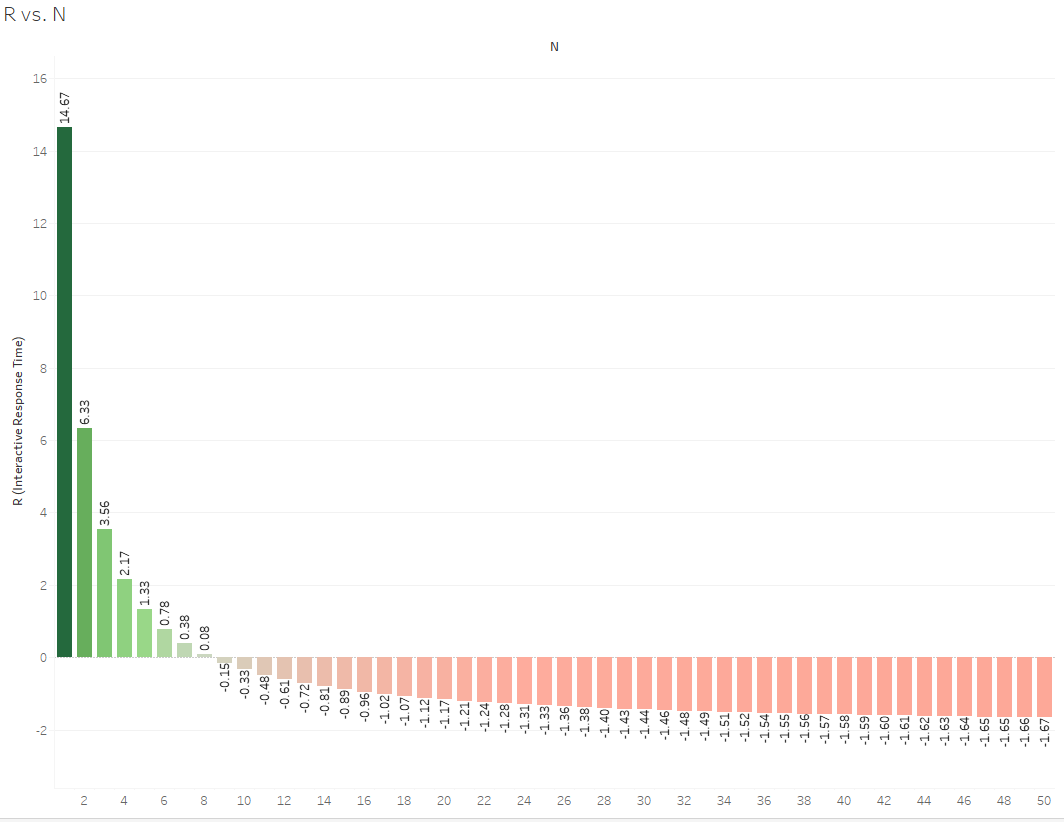


The findings ended up as they did due to the increased service demand with each iteration, causing higher amounts of throughput as the load test progressed.

The final visualization that I created was based around finding the Interactive Response Time (R) of the script and I used the following formula to work this out: R = (M / X0) – Z

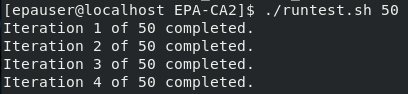


This shows that only the first 8 iterations of the load test have an Interactive Response Time of above 0, while the values eventually level out between -1 and -1.7 for the rest of the iterations. I believe this happened due to the very low values in the first few X0 values causing a high value for Interactive Response Time. This was eventually balanced as the value of X0 got closer to 50. The graph can be seen below:



**Notes on the Script**

When running this script from the command line the user must enter a number between 1 and 50 to run the script successfully. E.g: **./runtest.sh 50**. As the script is running through each iteration of the load test a message is displayed on the command line showing progress of the script. This is purely to keep the user aware of the progress of the load test.

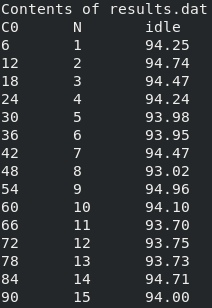


I ran the command **sudo yum install sysstat** to install sysstat on the Fedora VM. The script contains some input validation to ensure that the load test will only begin when a value between 1 and 50 is entered. The script also checks for the existence of both **results.dat** and **data.txt** before entering the for loop and will remove these files if they are found to ensure that each load test contains unique values each time it is run. The for loop iterates for as many times as the user has specified when they ran the file. The value **$1** is assigned to a variable called **m** which is used as the bounds for the loop as seen below:



A timeout of 2 seconds is set for each iteration of the loop and **mpstat** data is appended to the file **data.txt** to be used later the display the value of **idle**. The current word count of **data.txt** is stored in a variable called **c** which is then used to display the value of **C0** for each column in the script. Finally, the **awk** command is used to print the 12th column of data in the data.txt file which is the value for **idle**. This value is then appended to the results.dat file and a message is displayed stating that the iteration has completed.

When the loop finishes, the contents of results.dat are displayed to the user in the format specified in the CA Notes, as seen below.



**Notes on the VM**

The bash script was written and run within a Fedora Virtual Machine on the Linux Operating System.

**Virtual Machine Specs:**

* Name: Fedora 28
* Type: Linux
* Version: Fedora (64-bit)
* Memory: 2505 MB
* CPUs: 2
* Exec Cap: 100%

**Storage:**

* Type: Normal (VMDK)
* Virtual Size: 17.61 GB
* Actual Size: 9.91 GB
* Details: Dynamically Allocated Storage