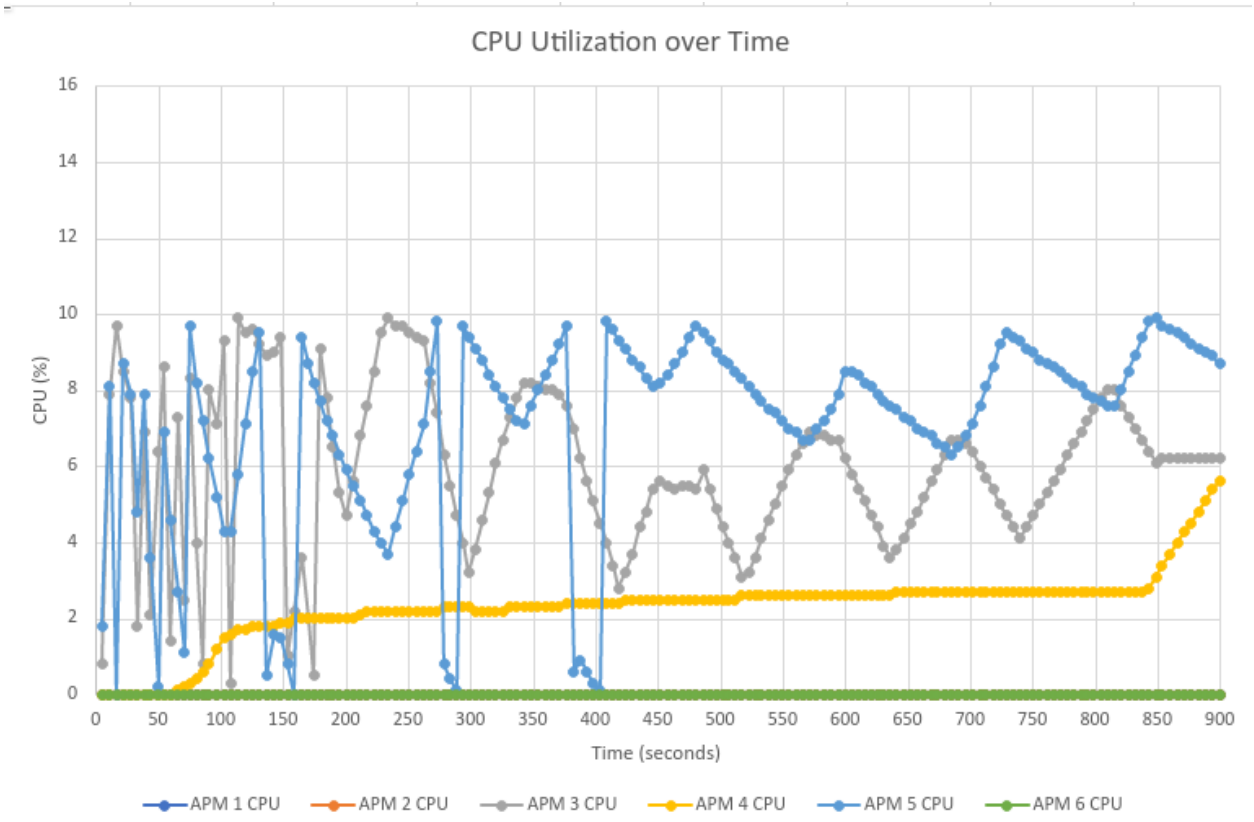
NSSA-220 Project 1: Application Performance Monitoring

Brayden Werner, Chase Killorin, Cade Gilbert

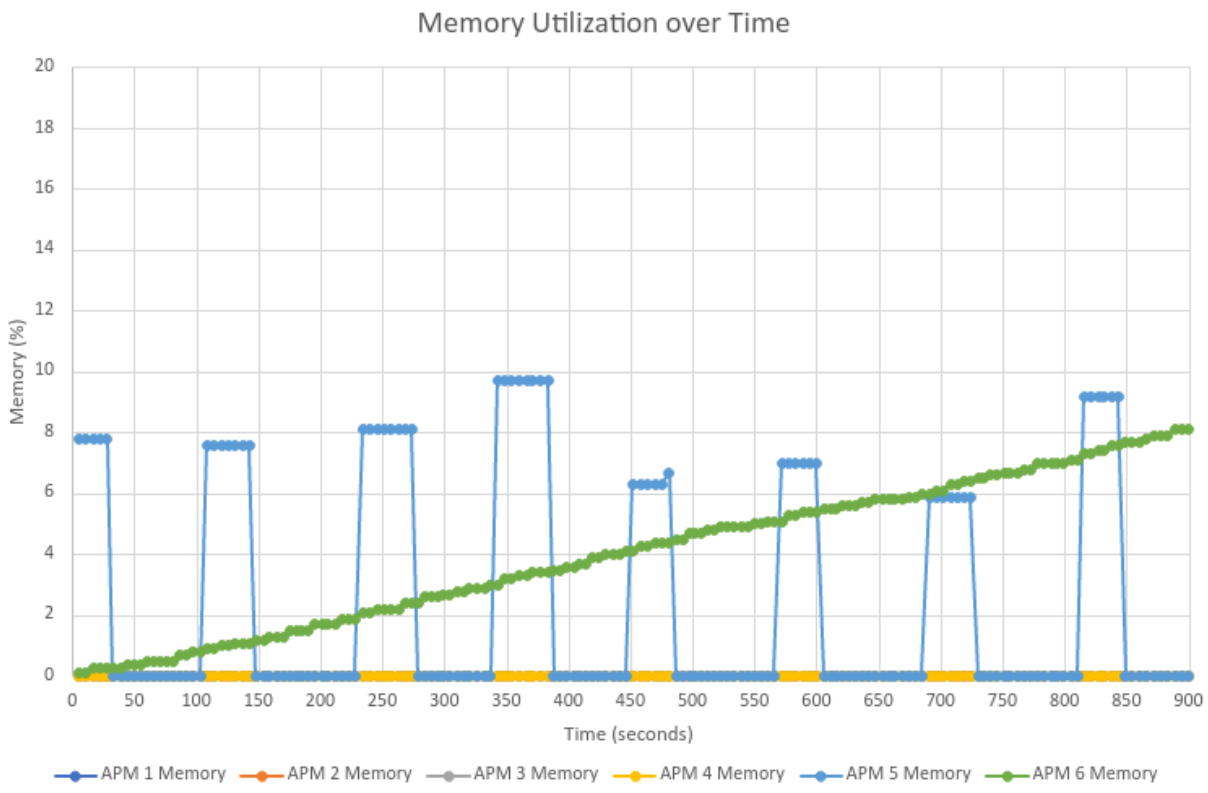
Introduction

The goal of this project was to automate Application Performance Monitoring within a bash script. It was also about working as a team and practicing communication and good team development skills. Furthermore, it was a learning experience to understand the in-depth system overview that bash gives you and to learn what is actually done in industry.

Process Level Metrics



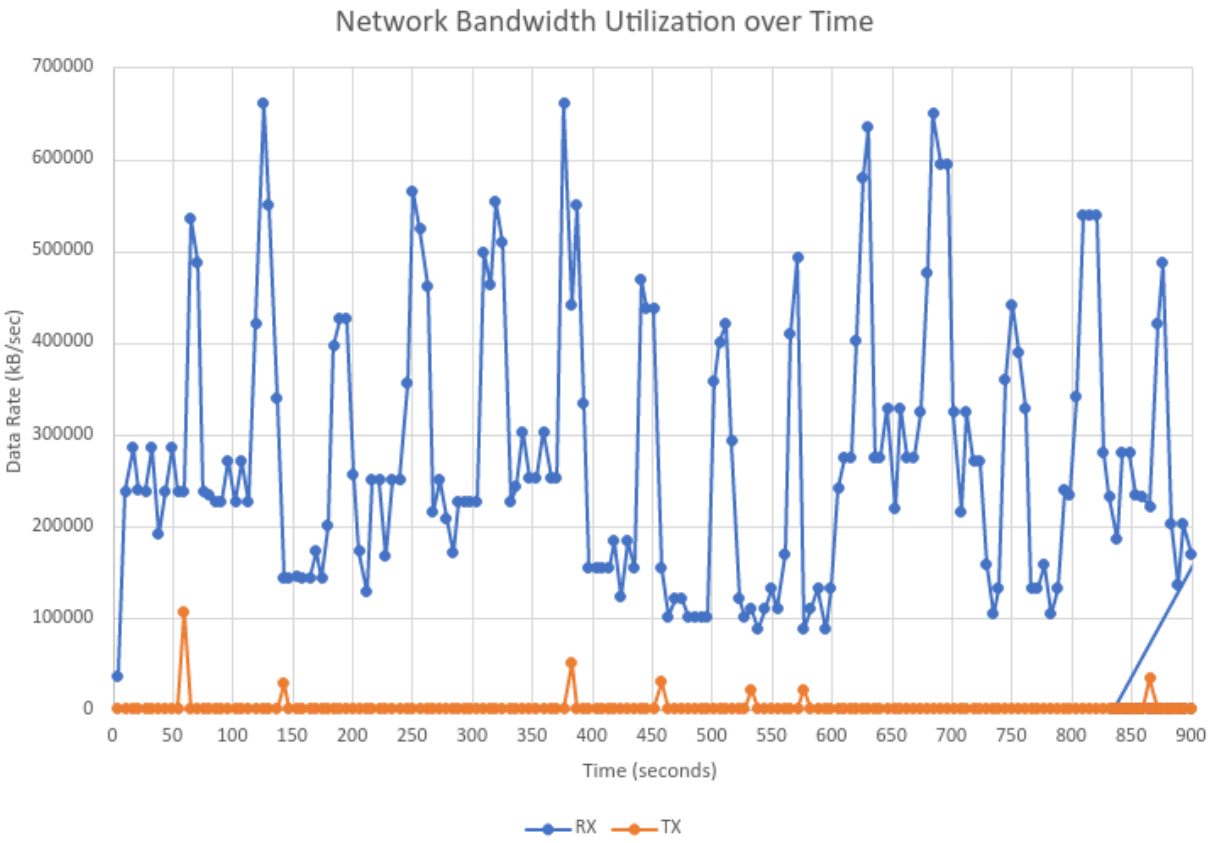
The total CPU utilization plot showed a varying influx in CPU usage over the first 6.5 minutes with major spiking as APM processes were allocated resources. During the latter 8.5 minutes utilization consistently ranged between 2% and 10% CPU utilization across APMs, with the two most active APMs: APM5 bouncing between 6.5% and 10% and APM3 bouncing between 2.5% and 4.5%. APMs 1, 2 and 6 had very little utilization and APM4 had a steady upward trend over the 15 minute span of time, spiking between minute 1 and 2 and then slowly climbing until minute 14 where it spiked again to its max value at nearly 6% usage.



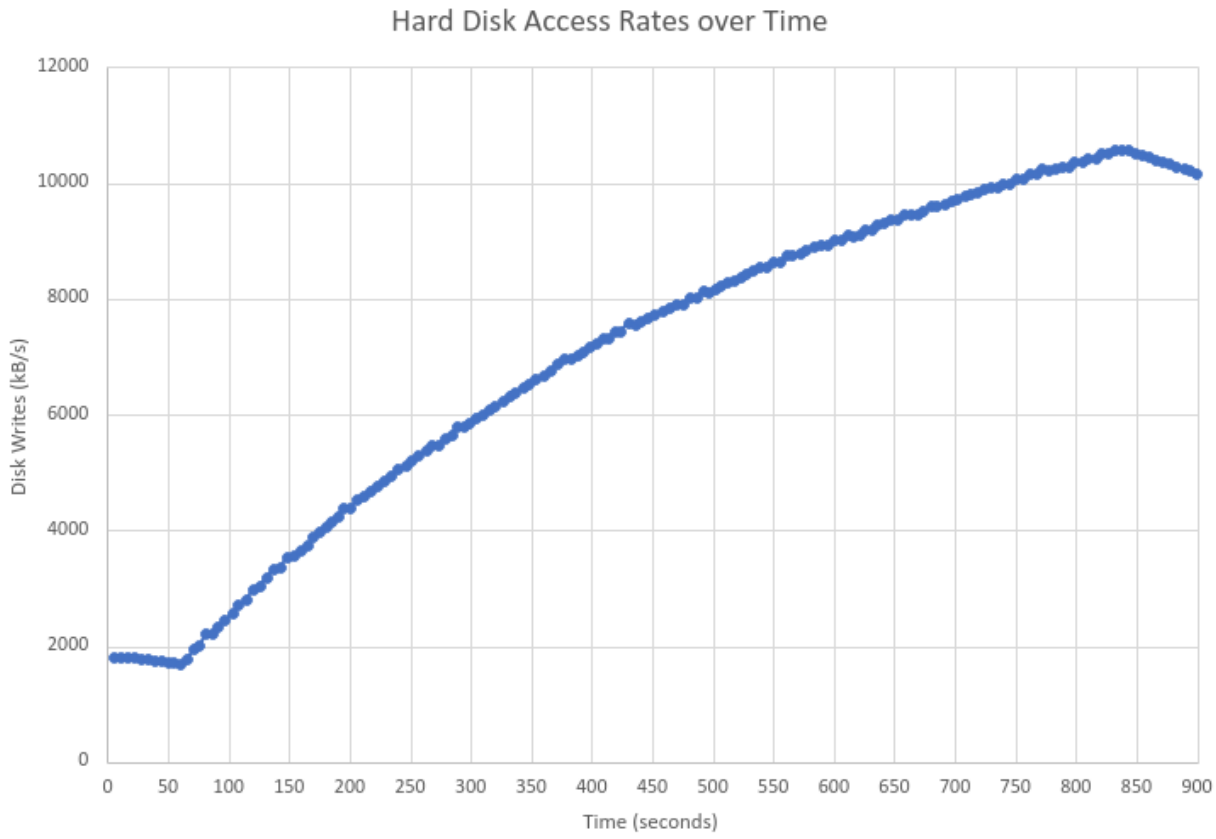
The total memory utilization plot shows that the average memory usage over the course of the test increased as APM6 gradually used more memory over its runtime, while APM5 spiked periodically around every minute or so to about 8-10% utilization. APMs 1,2, 3, and 4 barely used any memory at all. It seems due to the gradual increase in memory usage that APM6 may have had a memory leak causing it to never stop increasing, while APM5 may have just been a burst process that loaded a dataset and calculated with it for only a short period of time.

Generally speaking APM5 was the most taxing to the system overall with the highest burst memory usage and overall highest cpu footprint, which roughly line up with a program that loads a dataset into memory and processes it before offloading it, while APM3 seemed just to have a run-of-the-mill memory leak that caused its memory footprint to ballon steadily over time. APM3 ran the inverse to APM5 in processing, and APM4 could have been a recursive call, as it seems to increase steadily in processor usage over time.

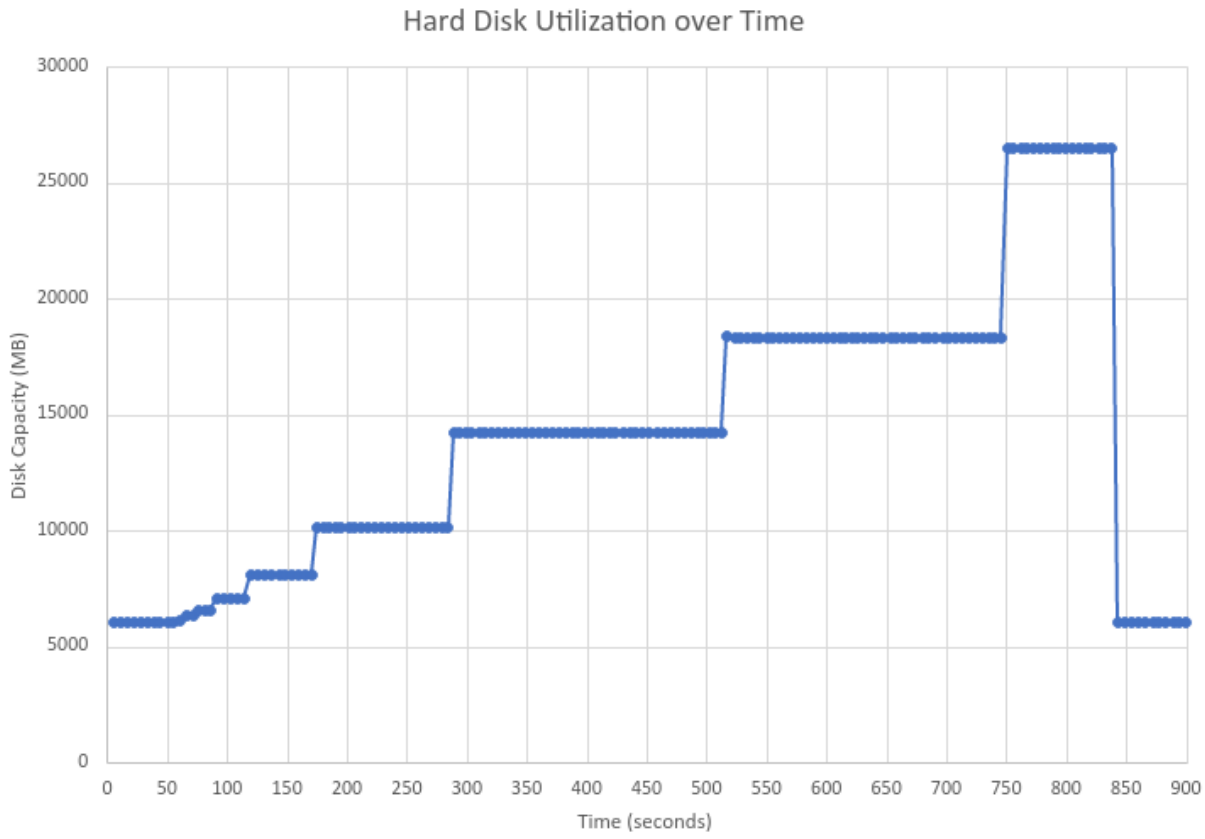
System Level Metrics



The network bandwidth plot seems to show that much of the bandwidth usage was utilized in receiving data from the APMs while barely any data was sent out. However, the amount that was sent out seems to roughly coincide with some spikes in received data, so we may be able to assume that the programs running may have been pinging and awaiting a larger response packet. On average the receive rate hung around 250000kB/sec ± 50000kB/sec while the receive rate only ever spiked to 100000kB/s at maximum and usually only to about 50000kB/sec.



Over time the hard disk writes increase steadily from what seems to be their baseline around 2000kB/s up to 10000kB/s. This is in line with the APMs slowly increasing in intensity over time and accessing the drive more as they write and read from it. This suggests that a process is slowly spawning more of its routine to access the drive, which could be inline with the displayed CPU usage of APM4, or the memory leak in APM6.



Over time the hard disk utilization increased proportionally to the access rates, this could imply a program procedurally writing more and more data from more and more sources, this usage started at about 5000MB and got all the way up to about 27000MB over the course of the test. However this increase was in large chunks that were about 5000MB in size, though this size was relative to the time into the test where at the beginning the chunks were exponentially smaller. This overall suggests that the hard drive access could be tied to APM4 or APM6 as its a gradual process.

Interestingly the TX and RX data rates seem relatively constant as compared to the hard disk rates, which could suggest a consistent download of data over time, or a networked process spawning child processes that all work on data as it is received, which could explain the increase in access rates over time but not the increase in RX and TX. Another reason this increase in access rates occurred could have been due to APM6 using the swap file more as its memory leak got worse. As was covered in my previous statement, the hard disk statuses were in relative lockstep and both increased gradually over time.

Summary and Lessons Learned

The CentOS VM had more than enough resources to handle the APMs. The system had enough CPU resources, memory, network capacity, and disk resources for the varying requirements for the different APMs. We learned working as a team that dividing up work and communicating with each other on the status of the project was really beneficial to getting everything done on time.