IBM i: When performance problems are caused by external factors

Back when I started my IT career 35 years ago, it was the “centralised” universe that originated from mainframe model. All core application codes ran in one and only one big iron that all users accessed with “dumb” terminals devoid of any GUI. Any attempt at problem solving in AS/400 systems was frequently straightforward and not much time consuming only because most cases were anything but elusive.

But contemporary IT infrastructure universe has evolved into a big onion for us to peel while troubleshooting. Many times, I find myself having to address a problem in multiple layers and it no longer surprises me when the cause(s) of the problem lies outside of IBM i server. In my long career, I found that many times when IBM i customers had problems, they almost always called IBM for help first and my colleagues (including trustworthy business partners) and I dutifully did our best to oblige, only to find in some cases that the causes of the problem were not in IBM i servers. But none of us could have known from the outset that this was the case because we have lived under the embrace of a multi-layer onion of IT world.

Let me share here with you two cases of such nature in the hope you gain some insight.

The first case was a customer who upgraded from POWER6 server with internal disk to POWER8 server with IBM FlashSystem V9000 SAN storage with flash disk technology. Within the first few months, the customer found it disappointing that the new server running their most important yearly batch process run time (with the heaviest workload to crunch) was only slightly faster than that on the old server. Such a two-generation server upgrade was quite a big boost in hardware performance and everyone expected corresponding boost in batch run time, but it was not to be. Another strange thing was that all other operations yielded expected level of improvement – data backup time, virtually all non-batch workload, remote SQL data access and such improved satisfactorily.

When I produced and analysed a set of IBM i PDI performance report charts, I found only the following two performance anomalies. All other charts exhibited no signs of performance issue. The first issue was on wait time in batch jobs.

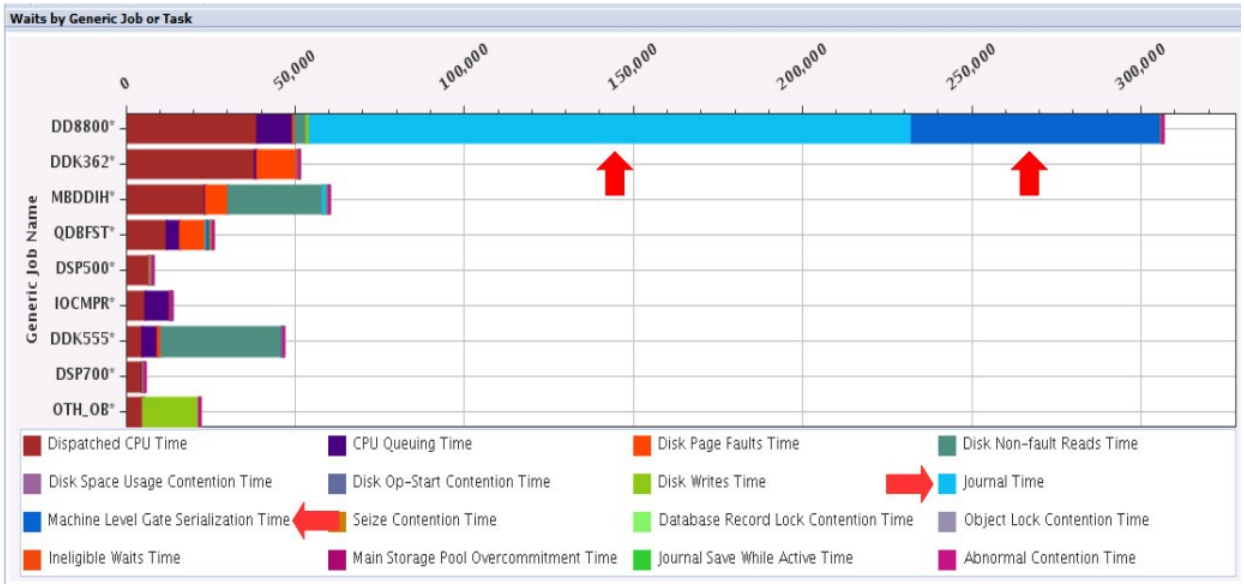

Figure 1: PDI chart for Wait by Generic Job or Task

In the Figure 1 above, batch jobs DD8800* encountered SUBSTANTIAL amount of Machine Level Gate Serialisation Time and DOMINANT amount of Journal Time. For IBM i users, I would ask them to regard this Machine Gate Serialisation wait time as similar to CPU Queuing wait time that I talked about in one of my previous articles. Its cause and solution are similar to CPU Queuing wait in some aspects. So, this issue was addressed in a straightforward way by reducing the number of concurrent batch jobs down to a proper amount (perhaps, by some 10-20% less).

To reduce Journal wait time, since the customer already bought and installed IBM i option named "HA Journal Performance", they just needed to identify all journal objects that covered all physical files used by DD8800* jobs and then use CHGJRN command to enable the parameter Journal Cache.

After taking actions above, the resulting run time improved but was not yet meeting the customer's requirement. So, we proceeded on to consider the next performance anomaly that I detected. This was degraded disk response time DURING the batch run period which was quite surprising considering the fact that SAN storage V9000 with flash disk was used! See Figure 2

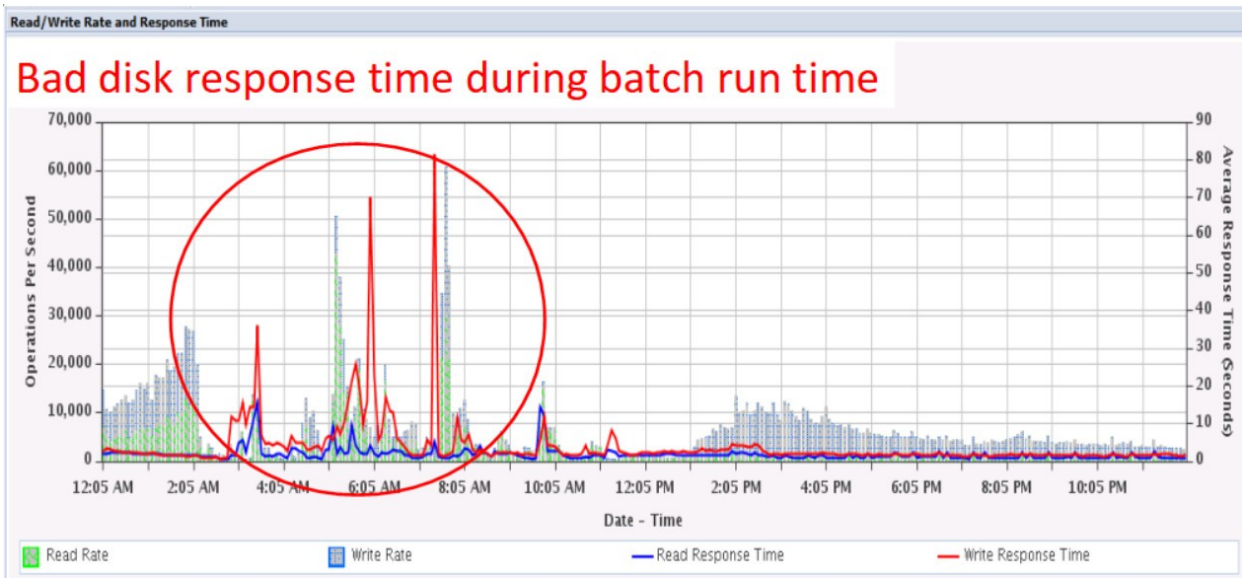

Figure 2: PDI chart for disk response time vs disk IOPS

You can see from the left half of the chart (batch run period) that disk read and write response times (blue and read lines in the red circle) are in a wide range of 5 to 80 ms! (The right axis is labelled "seconds" but this is a bug, the correct unit is ms.) The fluctuation span of response time is disappointingly high. This is perplexingly unexpected of a flash disk system since we can also see that the disk IOPS workload (blue and green bar graphs) during the bad response time period is NOT particularly high for flash disk. It stays below 10,000 IOPS most of the period which is low for flash disk with a few bursts that are still not very high.

Another aspect of this disk response time anomaly can be seen from the following PDI chart on Disk Overview by Disk Unit as shown below. Each horizontal bar represents average response time of each LUN over 24 hours. We have no need to see each bar clearly. The "forest" view is more informative here.

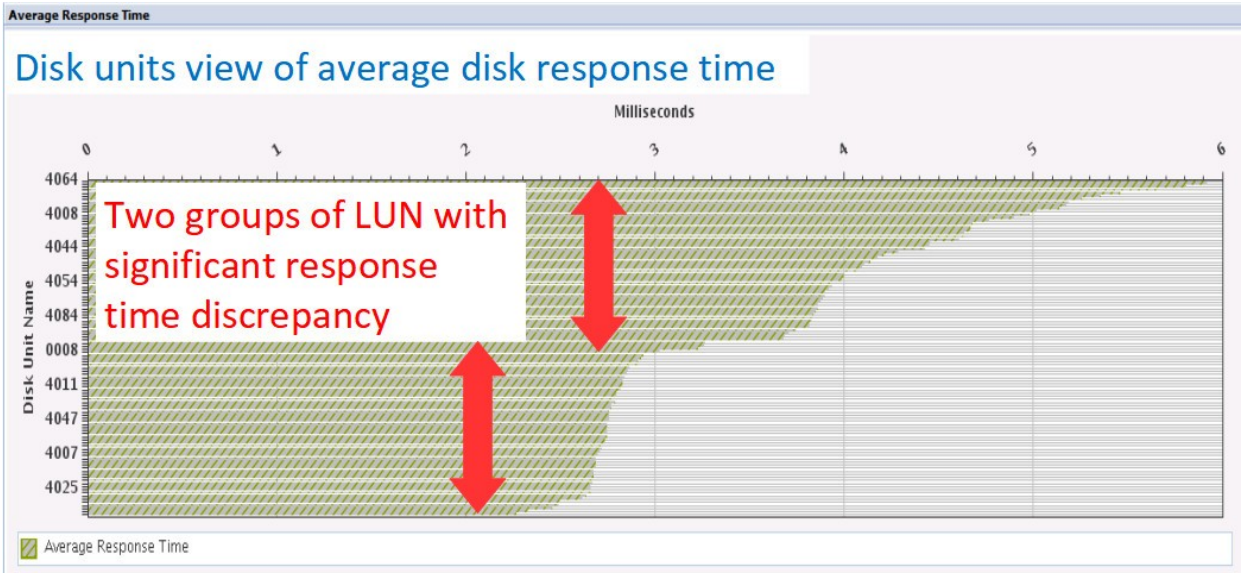

Figure 3: PDI chart on Disk Overview by Disk Unit

Figure 3 with the exhibited overall range of response time (6 ms and less) would look very good if it comes from spinning disk units. Although the response time discrepancy of about 1.5 ms between the two disk groups is not a big deal for spinning disk, no discrepancy at all is definitely the best. But since this chart is from LUNs over flash disk units, it does not look too bad yet but is somewhat worrying as to why the response time discrepancy exists for flash disks?

When I reported my finding to all parties, I was informed that the new POWER8 and V9000 servers were connected to an existing SAN switch used for some past years by the customer for connecting several existing Wintel servers to their SAN storage. The switch supported 16 Gbps fibre ports but its internal fabric ran only at 8 Gbps. Someone wondered if this contributed to the response time fluctuation to some degree?

Luckily, this was not a tough questions to answer. I asked for a disk response time report from V9000 itself of the same date and time as my PDI chart preceding the one above. The report came in a CSV-format file that listed V9000's response time in 5-minute intervals and this additional important piece of evidence confirmed to us all that the response time fluctuation corresponded to what I saw in the PDI chart. I also noticed that the response time from V9000 was somewhat lower than that from my PDI chart by about 10-20%. This made it clear that SAN switch had minor contribution to the problem at hand.

So, we had a clearly identified anomaly we needed to address. The question was what caused this fluctuating response time degradation in V9000?

Disk response time degrades when disk IO workload rises and there are two main disk IO workload types to consider for an analysis: IO per Second (IOPS) and read/write data transfer rate (MB/sec.). The rise of any of these IO workloads to high values causes disk response time to degrade in a varying degree depending on performance capacity of disk subsystem HW configuration and certain feature activation. The degradation is considered bad if it is consistently much higher than 5 ms.

From Figure 2, you can see from its leftmost side and the entire right half (left and right to the red circle) that response time does not degrade with rising IOPS rate (vertical green and blue bars). The picture is different from MB/s perspective. This rate (dark lines in the chart below) during batch run

period is on the high side as it exceeds 500 Mb/s in many instances and even goes beyond 1,000 MB/s a few times as shown in Figure 4 below.

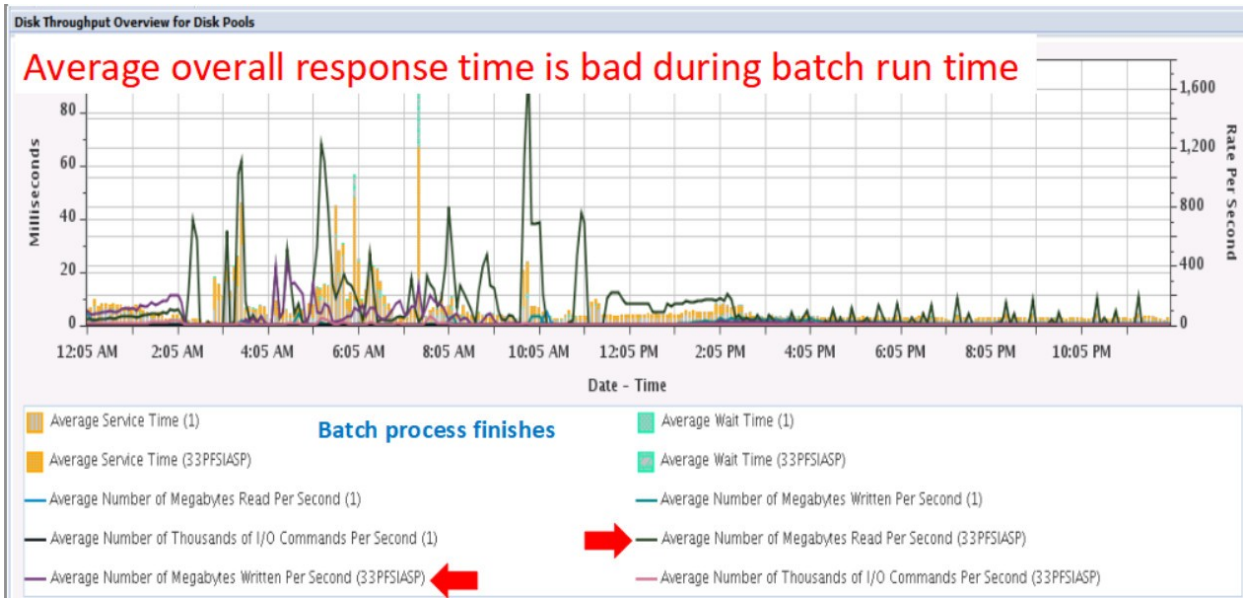

Figure 4: Disk response time vs read/write MB/s rate

According to my storage colleague, these high disk IO workloads were still under the capacity of V9000 to handle with ease. But then someone also mentioned that a feature called DRP (Data Reduction Pool) was activated in this V9000 which turned on both data compression and deduplication. This was a crucial piece of information for me because my past experience informed me that this kind of feature normally REDUCED the high-limit of performance capacity of the disk hardware in handling high IO workload with decent response time. I did a research and found in a few IBM information sources about implementing DRP that it took several considerations in designing DRP with decent performance.

At this point, I saw that I reached the end of my part of responsibility because the solution to this problem was in addressing the issue in V9000 about which I had insufficient knowledge nor experience. So, I relinquished the next steps to the storage specialist who designed this V9000 in order to rectify the wildly fluctuating and discrepancy in response times.

Later when I checked back with the case, I was informed that the problem was rectified to ensure decent and consistent response time for the customer's workload. It took some months for the rectification process and after it was finished, I saw good and more consistent disk response time under 1 ms with a degradation at high workload that did not go beyond 5 ms as shown in the PDI chart below. The customer also gained a satisfactory batch run time improvement as well.

In the left half of Figure 5 below, you can see MB/s workload (light blue line) rises to and beyond 1,000 MB/s in a few instances – the first one reaches almost 2,000 and stays in that range for about an hour - while response time (dark blue bars) manages to stay below 2 ms most of the time with few high bursts that are still under 5 ms. Such characteristic is highly desirable on any disk subsystem. I still do not like seeing the disk wait time (brown bars on top of dark blue ones) and suspect it may have to do with the SAN switch that I described earlier.

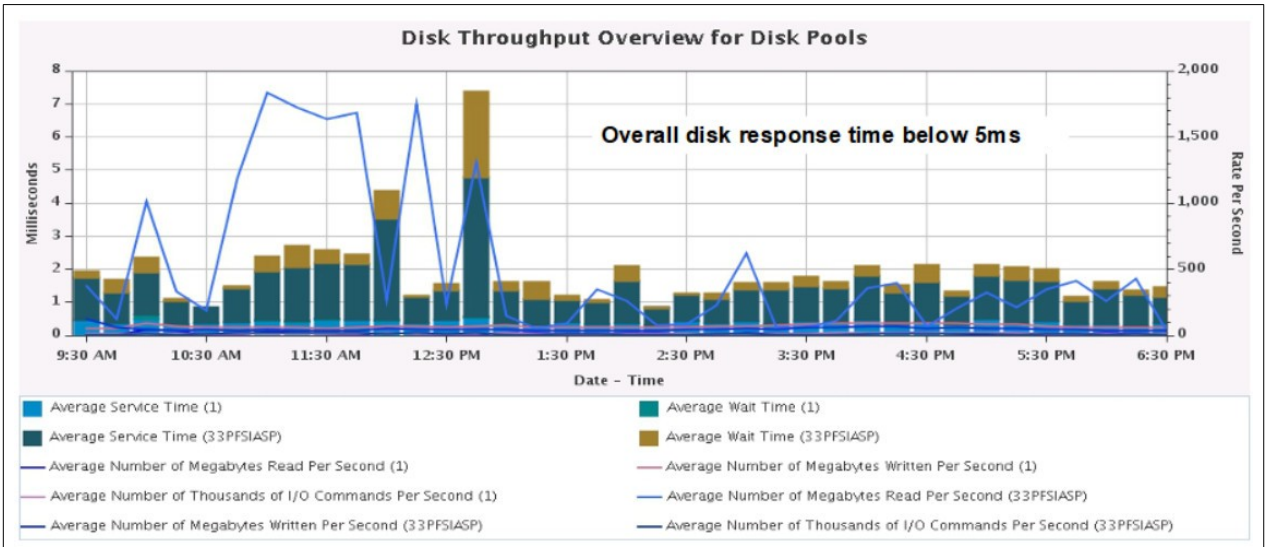

Figure 5: Disk response time vs read/write MB/s rate

The lesson learned from this case is that, even with flash disk technology, using sophisticated features that deliver more efficient use of disk space may come with a price of attenuated performance capacity of disk hardware if they are not designed and implemented with careful attention and sufficient relevant knowledge and experience. For production business workload environment, we need to be careful about using such sophisticated features if we know that good and consistent disk performance is our foremost priority. It is important to seek assurance from the SAN storage vendor to design SAN storage server in a way that ensures decent and consistent disk response time in fluctuating IO workload.

Lets move on to my second case study involving an external source of a performance problem in a complex IT universe.

A personal finance services company with its clients in the range of a million provides typical modern-day mobile/web services to clients. My customer runs a number of Wintel VMs (Virtual Machine) for web server front-end workload that interact with many back-end server jobs in IBM i (with 9 POWER8 CPU cores) that access the core application data to satisfy requests from the front-end.

When my customer upgraded their fibre LAN from 1 Gbps to 10 Gbps, a frustrating problem followed and attempts to correct the problem improved the situation somewhat but did not yield a decisive resolution. My customer added one more CPU core and extra memory, ISV back-end application team adjusted number of server jobs and relevant parameters such as max allowed connections per IP port and such, but the problem persisted in a significantly frustrating degree after a few months of collective effort.

The problem was that on the 25th and 26th of every month, the majority of my customer's clients accessed mobile/web services to process their monthly dues and many of the clients encountered such annoying problems as rejected initial connections or session freeze after established connections or connection drop during their transactions. Some had to try several times to get through the transactions. Naturally, many clients complained to the call centre. This problem also existed in the remaining days of each month but to a much lesser degree. Eventually, I was asked to come in to provide help.

When I initially obtained this description of the problem, it appeared to me this problem exhibited a strongly performance-related nature. (But I also realised at the back of my mind that if I had a hammer, everything would look like a nail!) Since this was a distributed application architecture, the key

question was where the bottleneck was? I started from the usual place, IBM i.

After obtaining sufficient background of the case, I spent many days checking many factors in IBM i such as job logs of the server jobs, low-level error logs, history logs, Db2 Plan Cache, and PDI performance reports but I did not find any things sufficiently suspicious. I found that IBM i server hardware + OS + DB2 appeared to be in decent, if not ideal, operational and performance health. This was when I told myself to look outside of IBM i as I was very certain the cause came from outside IBM i environment.

I started with what I could do which was relying on IBM i communication trace and Wireshark tool for basic network protocol analysis. The last time I used Wireshark was when it was new so many years ago, so I took some time to familiarise myself with the current one and I found two useful reports that helped me advance further on the path to a solution.

The first useful Wireshark report was Analyse --> Expert Information. Comparing the reports from a normal day against those from the problematic peak workload day, I saw that there were a lot of Connection Reset warnings on the peak day as shown below (the entry with red arrow and 37,526 incidents).

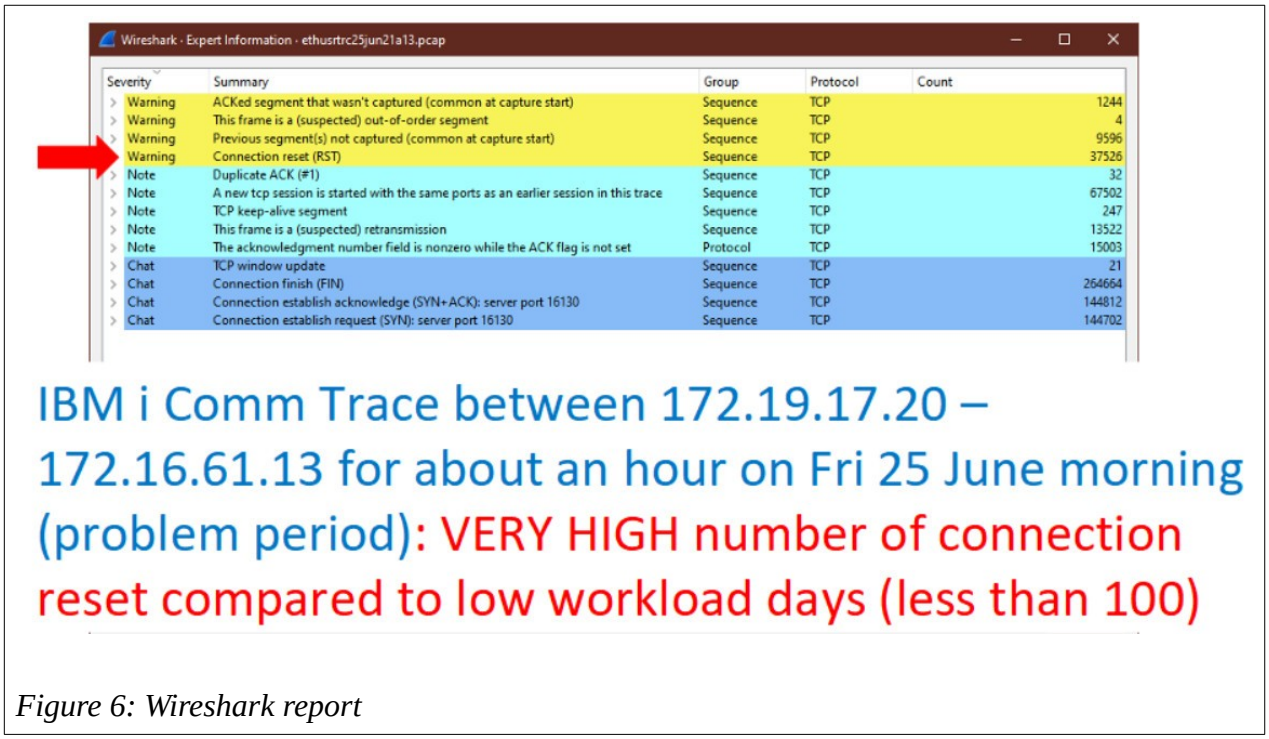

Figure 6: Wireshark report

Further drilling down revealed to me that ALL Connection Resets were sent from relevant IP sockets used by web services server jobs in IBM i to their corresponding IP sockets used by front-end VMs for the troubled transactions. A deeper analysis help from a network protocol expert uncovered the cause that these resets were all due to back-end server jobs receiving corrupted data packets (incorrect data length information) from front-end VMs. The responding connection resets from back-end server jobs was therefore normally expected. This produced a question of why some packets from front-end were corrupted? The higher the transactions workload, the more the corrupted packets.

For a few days, a search back into my deep past memory hit an incident of similar nature where I was involved in helping a case in which SAP ECC on i was implemented with two iSeries servers, one as DB server and the other as Application server. This was to handle very high workload from a thousand users who complained about randomly failed transactions at peak workload periods which occurred almost everyday during a certain time period of a day. Although I detected no serious performance

issue within both iSeries servers, network protocol analysis from an expert revealed many corrupted data frames sent from App server. The expert told us he detected what was called TCP buffer overrun and advised that we increase the size of TCP send and receive buffers in both servers to about 1 MB each (it was about 200 KB at the time of problem) and this did happily solve the problem. TCP/IP buffer "overrun" is the result of too small a buffer size at a very high network traffic.

Based on this memory of past experience, I suggested the customer increase TCP buffers on both IBM i and Wintel VMs to 500KB but it turned out that, for Wintel server, the buffer was adjusted at network adapter level and my customer could not find anyone who knew how to do this.

Another feedback from the network expert was that App severs opened and closed socket connections for each and every transaction which was a wasteful practice as it incurred protocol overhead for each transaction which added latency to each connection's performance. It increased network traffic volume as well. A well-known approach to reduce such network protocol overhead was to create a pool of "persistent" connections in each front-end VM and allocate and deallocate each pooled connection as needed. This eliminated connection opening and closing protocol exchanges and reduced overhead and latency. Since this approach took time and effort to implement, the customer wanted to pay attention to it later after the current problem was properly solved.

So, the problem persisted. Luckily, I noticed another Wireshark report that looked potentially promising which was Statistics --> Endpoints report as shown in Figure 7.

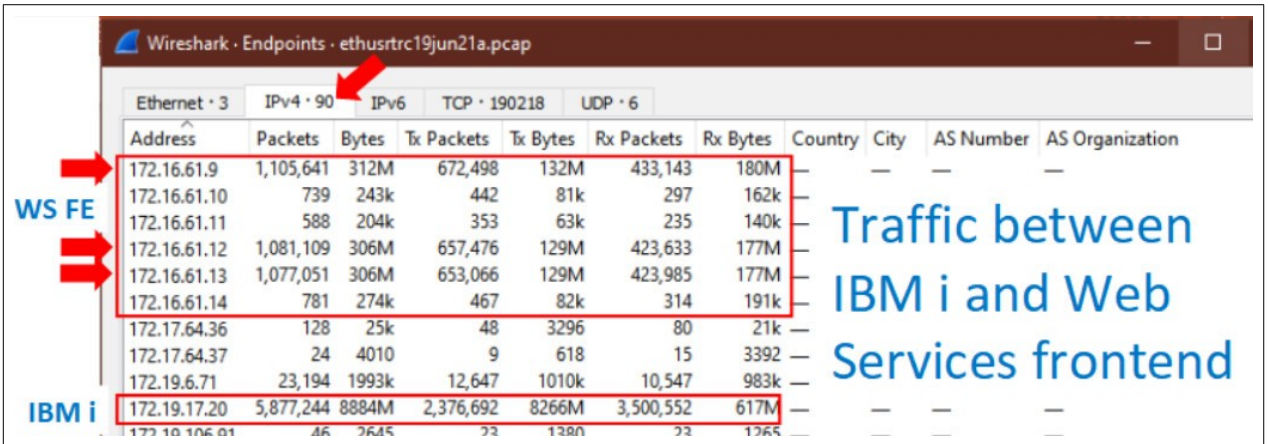

Figure 7: Endpoints report

Based on a sample result above, I discussed this report with my customer and was informed that they expected that all 6 front-end VMs carry incoming workload equally (IP addresses ending 9 to 14 in the report) but the report clearly showed that this was not the case, to my customer's dismay. As seen from the report, only 3 front-ends (address 9, 12, 13) carried most of the transaction workload while the remaining 3 carries very little (address 10, 11,14). This was immediately rectified by my customer and subsequent endpoints report verified the desired workload distribution.

With actions taken as described so far, we waited until the following peak day and reviewed Wireshark reports again and the result was that the count of Connection Reset reduces from the original 37,000 range down to about 9,000 BUT all of us wanted a zero count. I proposed that a sensible way to eliminate Connection Reset was to ensure that all front-end VMs did not carry overwhelming workload that interfered with TCP/IP protocol exchanges and proposed the customer increase the number of front-end VMs to at least 12 from the current 6 and the customer agreed immediately and increased to 15 VMs. The result was that the following peak day went by WITHOUT any problem for the first time in many months, to everyone's joyful relief.

The last hurdle the customer wanted to address was to reduce network protocol overhead using a pool of persistent connections. They set up a new test front-end VM to implement this method and found one last problem for me.

Every morning when they started to test persistent connections, they found that all “supposedly persistent” connections were all disconnected. They restarted the connection pool and did the test for the day only to find the same connection drop again the following morning.

When they consulted me, it reminded me of another past experience I encountered a few times before which had to do with Firewall that stood between front-end and back-end servers. One thing a Firewall is designed to do is to drop any connections that stay idle for too long using a parameter called Idle Connection Time-out which normally is set in a range of 5-10 minutes by default. On the server side, one TCP/IP feature called Keep-alive is available to prevent a persistent connection from idling for too long and it is control by Keep-alive Timer that is normally set at longer than 10 minutes by default – 120 minutes in IBM i.

I explained this to the customer and asked them to try increasing Firewall's Idle Connection Time-out to a value longer than 10 minutes, say, 30 or 60 minutes and set TCP Keep-alive Timer to some 5 minutes shorter. My customer took the action and informed me later that the problem was solved. I considered my contribution to this case at its end and the customer obliged.

In summary, this problem had its root cause in UNBALANCED and OVERWHELMING workload at the front-end VMs that interfered with their TCP/IP's underlying mechanism but somehow the customer overlooked this part and reached IBM for help and my colleagues and I accommodated the request and gained new experience in the process, not to mention gaining back a happy customer also.

Multiple layers of the infrastructure onion made the troubleshooting process time consuming and we could not address an issue in a layer and had to move on to the next one in which we eventually succeeded. The utility that Wireshark tool provided in the process was undeniably crucial. All is well that ends well, according to the Bard (Shakespeare).

I hope you agree with me now as I said earlier that troubleshooting in IT universe these days is dealing with a multi-layer onion of IT infrastructure. I would say those who appreciate this fact and keep an open mind and curiosity and act accordingly as a team would well be on the right path to the desired destination.

Satid Singkorapoom has 31-year experience as an IBM i technical specialist since it was called AS/400 + OS/400. His areas of expertise are general IBM i system administration and performance of server HW, OS, and Database. He also has an acquired taste in troubleshooting problems for his IBM i customers in ASEAN geography.