IBM i: Using IBM i PDI charts to answer performance questions

Satid Singkorapoom

In my long career as an IBM i performance specialist, it was natural for people to ask me performance-related question of various kinds. But when I provided answers in many past cases, it appeared to me some of them might not understand my answers well when I did not provide supporting evidence due to the lack of a performance report tool that could produce easy-to-understand explanatory information. The situation changed for the better with the availability of IBM i Performance Data Investigator (PDI) tool as of release 6.1. PDI tool succinctly illustrates the meaning of the saying "A picture is worth a thousand words". Let's explore some cases of PDI tool's persuasive power to provide clear performance-related answers.

Why do I need to use more CPU power?

An SME customer runs Java-based core application in a 1-core POWER8 server. There is only one main java core application job with 1,200 threads. The customer encounters persistently slow application performance at every month-end period, which they know is the peak workload for the entire month. The customer observes that during the duration that users complain about long response time, WRKACTJOB command shows high overall CPU % Busy but it reaches 100% only sporadically and never lingers there too long. The question is whether they need to activate more CPU and why?

I produce PDI performance report and browse through charts on CPU and jobs, disk performance, memory faulting, wait times, and a few more and find that CPU never reaches 100% persistently, disk performance is consistently decent, memory faulting rate is never abnormally high, but the Figure 1 - "Wait Overview" chart does not look good as shown here.

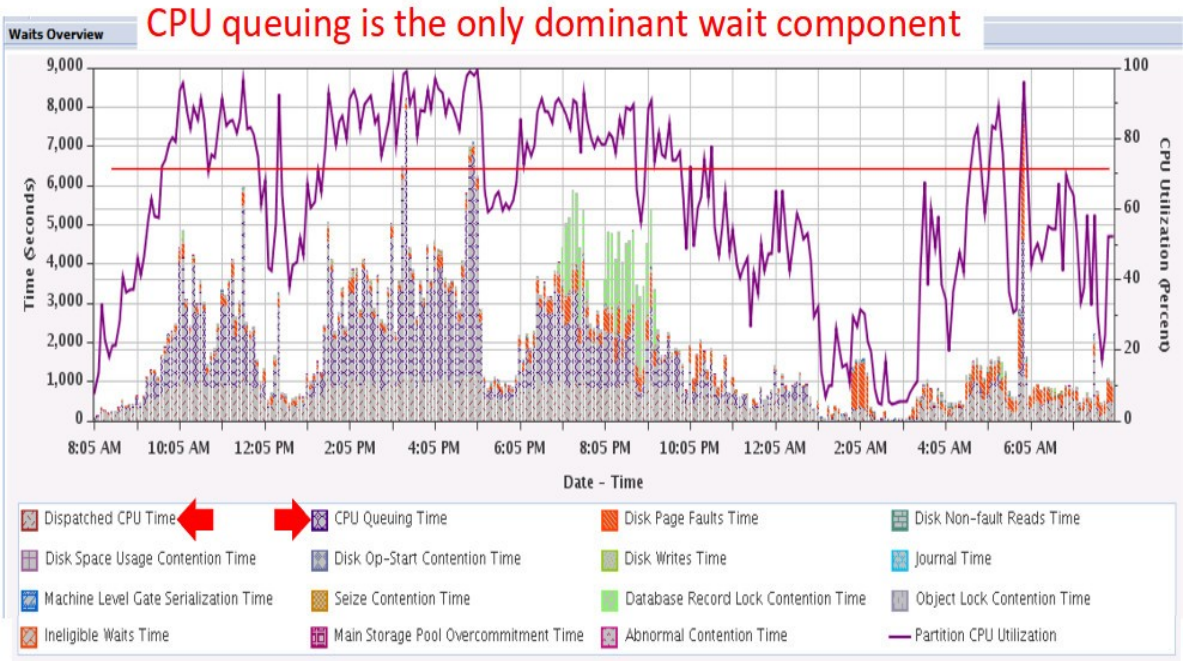

Figure 1: Wait overview chart

You can see that CPU Queuing wait time exists in substantial to overwhelming proportion against Dispatched CPU Time whenever CPU % Busy reaches 70% or higher (the red horizontal line in the

chart). CPU Queuing time is at the largest value when CPU % Busy is 100% at some instances but it is very small when CPU % Busy is lower than 70%.

The “Wait by Job or Task chart” reveals more that the core java application job appears to suffer the dominant proportion of CPU Queuing wait as in Figure 2.

Note:	Since the customer confirms to me that there is only one core application job, I use Wait by Job or Task chart rather than Wait by Generic Job or Task but both display the same data for this core job.

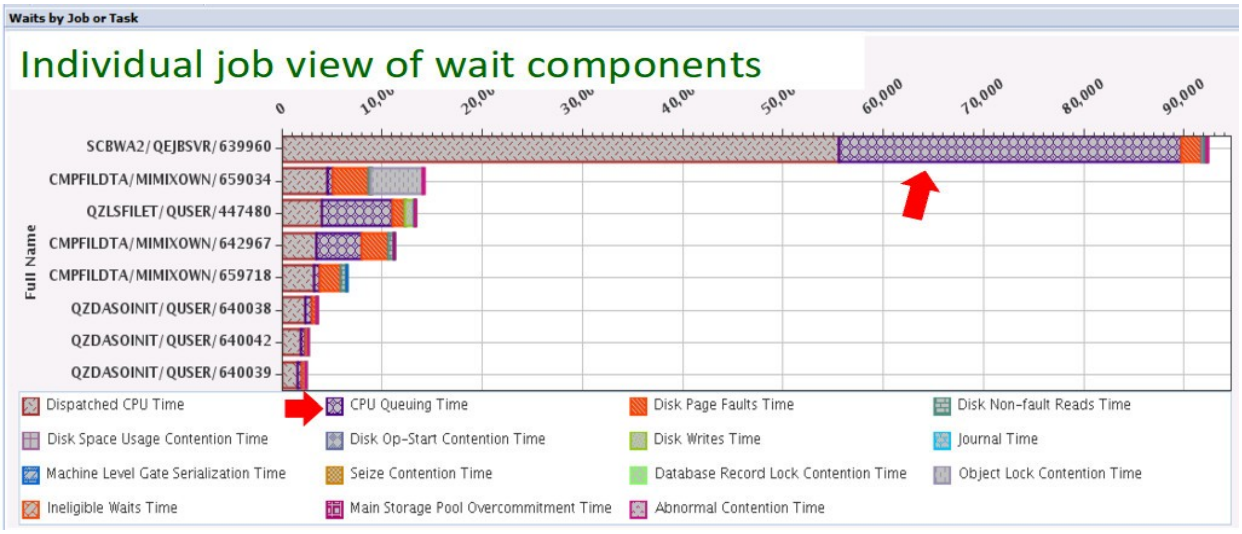

Figure 2: Wait by job or task chart

At this point, the customer has seen that there is no performance bottleneck in all other hardware and operating system components and I point out that this is a straightforward analysis that they need to activate at least one more CPU core to reduce or eliminate CPU Queuing wait time. The customer agrees and here are the resulting month-end Wait Time charts after the addition of the second CPU core in Figure 3 and Figure 4.

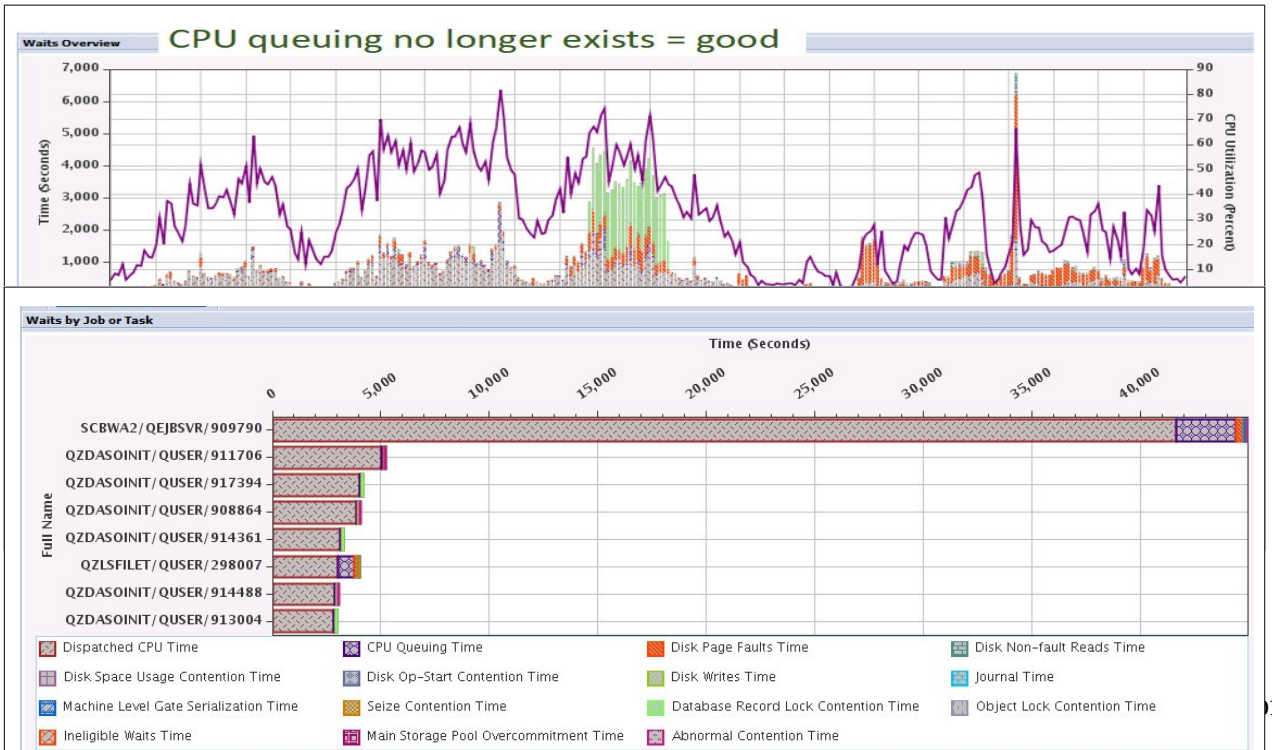

Figure 4: Wait by job or task chart with an additional core

CPU Queuing Wait has almost completely gone and users no longer complain about long response time. The overall CPU % Busy also reduces noticeably. The Database Record Lock Contention Time (light green bars in the Wait Overview chart) is addressed by creation of database indexes which I will discuss later in this article.

Do I need to add more memory?

This has been a tough question to answer for so long. But in early 2020, there was an enhancement to PDI tool (through PTFs for IBM i 7.3 and 7.4) which added a new set of charts to "Memory" category in which you can look at four new charts-: "Memory Usage by Pool"; "Memory Usage Peak by Pool"; "Memory Usage Minimum by Pool"; and "Memory Available by Pool". The last one provides the answer to you on whether you need to add more memory or not and which memory pool to allocate the additional memory. "Available" here means unused, that is if you see a sizeable amount in the chart, it means you still have unused memory in a particular pool, which can be moved to another pool that may need more. Let's look at Figure 5.

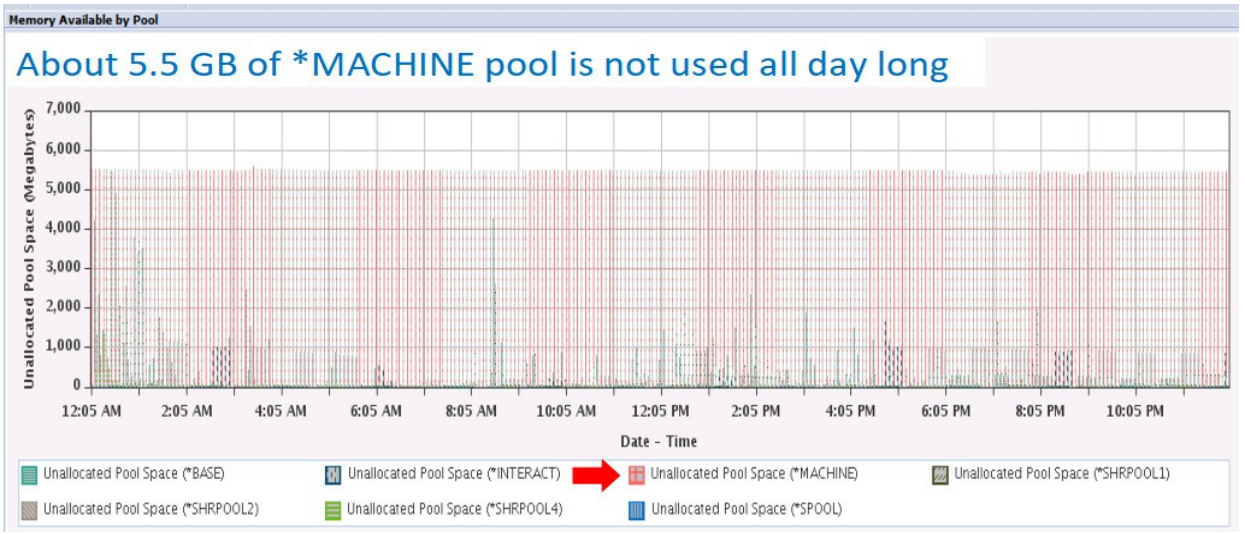

Figure 5: Memory available by Pool chart

In this 24-hour timeline chart "Memory Available by Pool", you can see that *MACHINE pool (bright pink colour bars) has about 5,500 MB (5.5GB) of memory lying around unused all day long!

To answer which memory pool you should move this excessive memory to, here we employ "evidence of absence" to make this decision. This means the bar graph representing a memory pool that uses up all its memory will not appear in this chart at all if this happens all day long. Or if it uses up its memory for some periods in a day, you should see its bar graph just here and there in the entire chart and its bars may not appear high in value which indicates modest amount of memory being unused. A memory pool that exhibits such a trait is a good candidate to receive the memory.

Please remember that you should look at this "Memory Available by Pool" chart on several peak workload days to ensure whether some pools have significant unused memory left while other pools need more on a regular basis or not. If this does not appear to be regular, you may not want to adjust memory pool allocation yet. It may also be the sign that IBM i Automatic Performance Adjuster (QPFRADJ system value) is working in moving the unused memory around for the pool in need. Only when you see excess memory in the same pool(s) (like the sample above) and absence of unused

memory of other specific pool(s) over several peak workload days should you adjust the pool size. You may also want to learn to use WRKSHRPOOL command to limit min and max size of the pools based on what you see in this chart.

I think that, in a desired case of sufficient memory for all pools, you want to see that all pools display modest amount of unused memory all day long in the chart of peak workload days. But if you see a blank chart, what does this mean? First I ask you to look at this over several days to confirm the case. When it is confirmed, it means the time for you to add more memory to your system urgently.

I have multiple disk pools (ASP) in my server. How does each perform?

When we investigate performance cases, the customers may have more than one disk pool (ASP) in their servers. When we look at disk performance from PDI charts, some display only for System disk pool while others display values that are the average or aggregation from all pools. There is a few other charts that display each pool's performance for a comparative view. This last type of charts helps provide answer to the question. One chart we should always look at in such cases is "Disk Overview for Disk Pool" as shown in Figure 6. (Keep in mind that there is another chart named "Disk Overview **by** Disk Pool" which is not what we want here.)

> Note: Another chart that comparatively displays disk service time and disk wait time by each disk pool is "Disk Throughput Overview for Disk Pool". It is another useful chart for analysing disk performance but it is also a more complex chart to analyse than Figure 6.

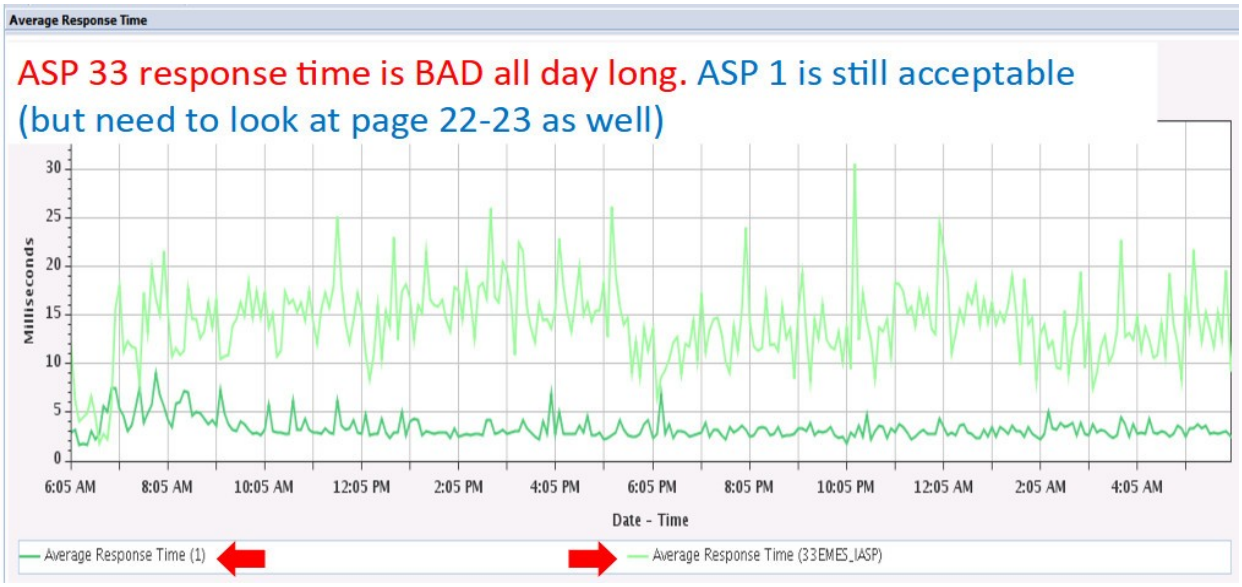

Figure 6: Disk overview for disk pool

This 24-hour comparative chart tells us right away that overall average response time of ASP 33 (EMES_IASP – the light green line) is quite bad all day long (much higher than 5 ms.) while ASP 1 (System ASP) performs mostly in a decent range (5 ms. or lower). So, the solution is that we need to improve the hardware of ASP 33 to restore its decent performance. Case close.

But in this particular customer case, when I look at Wait Overview and Wait by Generic Job or Task charts (Figure 7 and Figure 8), I see the following results as I mention in Figure 6).

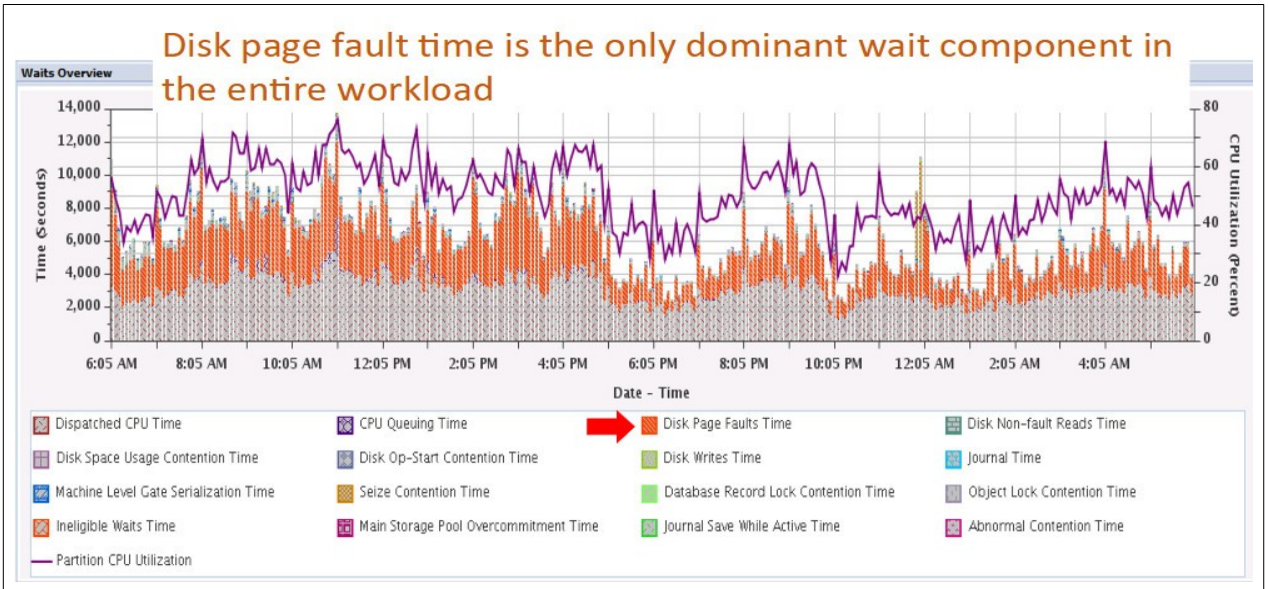

Figure 7: Wait overview chart

The proportion of Disk Page Fault Time wait above is not a lot nor overwhelming (comparative to Dispatched CPU Time). This wait time is a result of memory faulting. Each faulting causes disk read or write and therefore this type of wait time. Figure 8 shows interesting information.

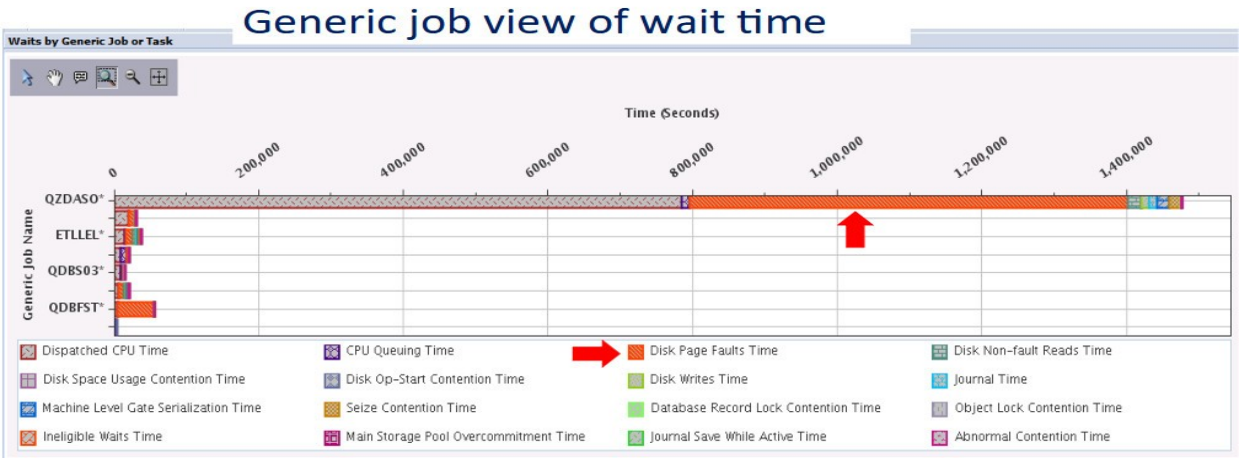

Figure 8: Wait by generic job or task chart

It turns out that QZDASO* jobs (QZDASOINIT) accumulate almost all Disk Page Fault Time in the entire 24 hours. This customer runs IBM i as a database server responding to remote SQL requests (ODBC/JDBC) from many web-tier servers. So, QZDASOINIT jobs carry the main workload as seen from the chart. The proportion of Disk Page Fault Time is not overwhelming against Dispatched CPU Time but it is substantial. If we can reduce this dominant wait time, we can improve overall performance of this client-server application.

All memory faulting happens in ASP 1 only. Although we saw earlier that ASP 1 provided decent response time, if we can reduce overall memory faulting rate in all QZDASOINIT jobs, it will reduce overall Disk Page Fault Time. Since this is a case of SQL workload, we can make good use of tools in DB2 for i to deliver this improvement. How?

When we deal with SQL workload (and Query for i), one popular cause of high (and unnecessary) memory faulting rate is that SQL engine resorts to table scans too much due to a serious lack of useful indexes on tables the workload accesses. Here scan means all rows in a table are read. The larger the tables, the higher the memory faulting as a result of table scans. In many cases, a table is scanned just to retrieve only one row. What a wasteful operation! You can prove this using Visual Explain on Plan Cache snapshot data.

I personally create indexes for tables larger than 100MB. Collectively, index probes (directly reading index entries of interest) and scans cause much lower memory faulting than table scans. So, it is worthwhile to be mindful of this fact and use DB2 tools to identify and create useful indexes. The main tools we use to achieve this goal are Plan Cache snapshot, Visual Explain, Index Advisor, and Index Condenser. These are available through Navigator for i GUI tool. Once we create sufficient number of useful indexes, the performance result is seen in Figure 9.

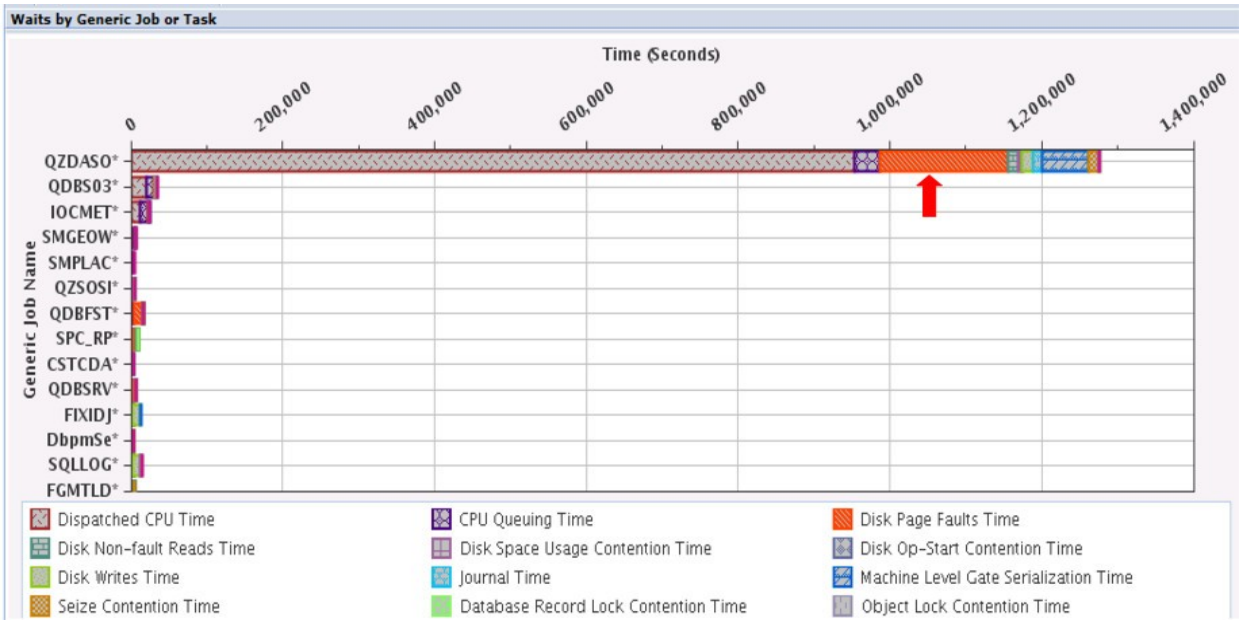

Figure 9: Wait by generic job or task chart

Compare to the earlier chart (all charts in this section are from the same customer case), you can notice that Disk Page Fault Time reduces by about 70% and users report workload performance improvement.

Do I have workload growth or reduction in my server?

In my long experience, it turns out that I frequently ask my customers this question rather than they ask themselves. PDI tool has one useful chart I would like you to be aware of, it is called “Resource Utilisation Overview”. Actually, this PDI item contains two charts and it is the second chart I’m talking about here and it looks like Figure 10

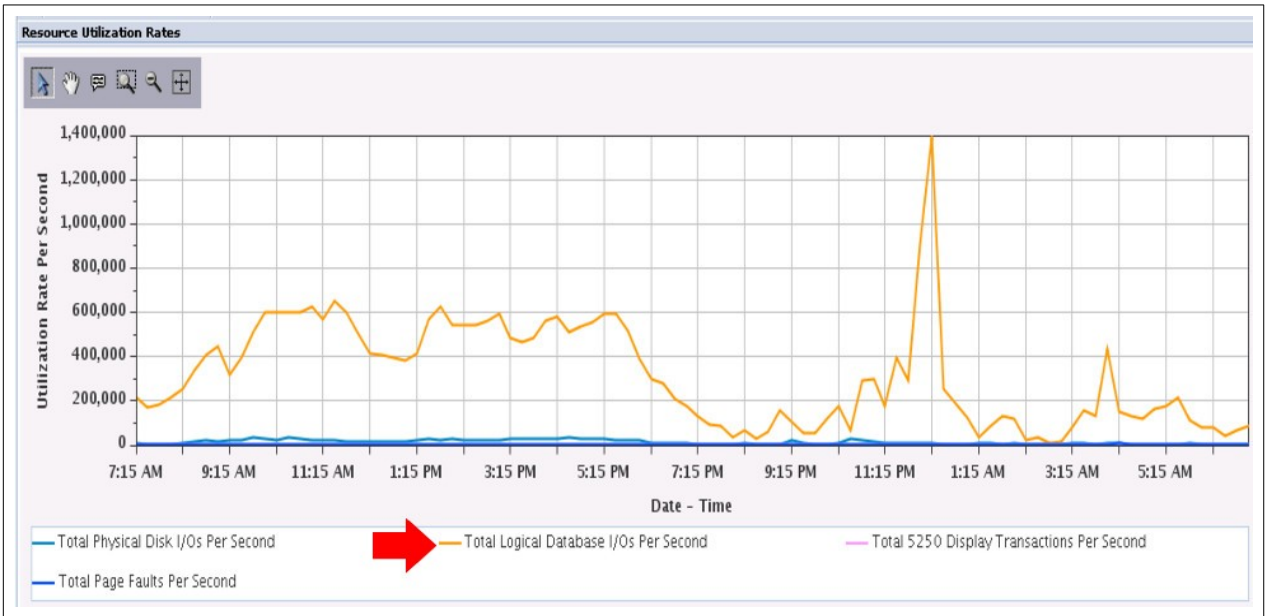

Figure 10: Second chart in Resource Utilisation overview

In this chart, I see that the line graph for Total Logical Database I/O Per Second (orange line) is a good representation for typical business processing workload as it heavily accesses database. If you run only web serving workload in IBM i (that mainly accesses stream files instead), this chart is not applicable. You may use Dispatched CPU Time in Wait Overview chart instead as web server is CPU intensive workload.

The usefulness of this chart is when you look at two such charts from different servers or time frames to have a comparative workload view. A good case to use this chart is when you are about to upgrade your Power server to a new model. Comparing these peak workload period charts from your old server against the new one can give you meaningful comparison, but you also need to re-adopt an ancient concept in geometry from your school days on calculating the area under a graph, I kid you not!

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.