IBM i: Dealing with CPU queuing wait time

When we drive our car to a destination, I figure all of us naturally hope that our cars never get a red light nor stuck in a slow traffic. We all hate waiting immobile in the traffic. I also figure that you are aware, fully or subtly, that active jobs in any computer system can encounter wait as well and there are many types of wait categorised by the IBM i developer team.

In this article, let's look at CPU Queuing wait time and how we can interpret and address it in a sensible way to resolve a performance issue. In this article, I hope to provide you with a useful approach to what is called wait time analysis using a gloriously useful performance report tool.

First, how do we look at this CPU Queuing wait time? As of IBM i release 6.1, a new built-in GUI performance report tool named IBM i Performance Data Investigator (PDI) was delivered under the browser-based Navigator for i system management GUI tool[1]. You can use PDI to display many useful performance data charts that help a knowledgeable person analyse IBM i performance health in many aspects such as CPU utilisation, disk IO workload and response time, memory faulting, wait times, etc. I will try to enable you to be knowledgeable in interpreting CPU Queuing wait time in this article, and a few more PDI charts in my subsequent articles as well.

> Note: In 2021, Navigator for i tool for system administration tasks was subjected to vulnerability relating to Log4j V1.x component and IBM subsequently delivered a new version of Navigator for i that no longer used Log4j V1.x. Please consult this IBM security bulletin on how to use the new tool.[2]

One very useful group of PDI charts is the “Wait” category and CPU Queuing is in this group. Wait time data was introduced in IBM i 5.4 but there is no PDI tool to display it at this release – you directly query the relevant performance data file instead to look at this wait time category but this is somewhat complicated to do. From my long experience in analysing IBM i performance, I can gladly say that wait time data are praiseworthy in helping me efficiently analyse performance issue since IBM i 6.1.

First of all, please keep in mind that you always need to look at two charts under Wait category which are “Waits Overview” and “Wait by Generic Job or Task” (or Wait by Job or Task – but the “generic” chart is more useful in my experience). Let's look at examples of these two charts. Figure 1 is a sample of Wait Overview.

1 https://developer.ibm.com/tutorials/ibm-i-performance-data-investigator/
2 https://www.ibm.com/support/pages/security-bulletin-ibm-i-components-are-affected-cve-2021-4104-log4j-version-1x

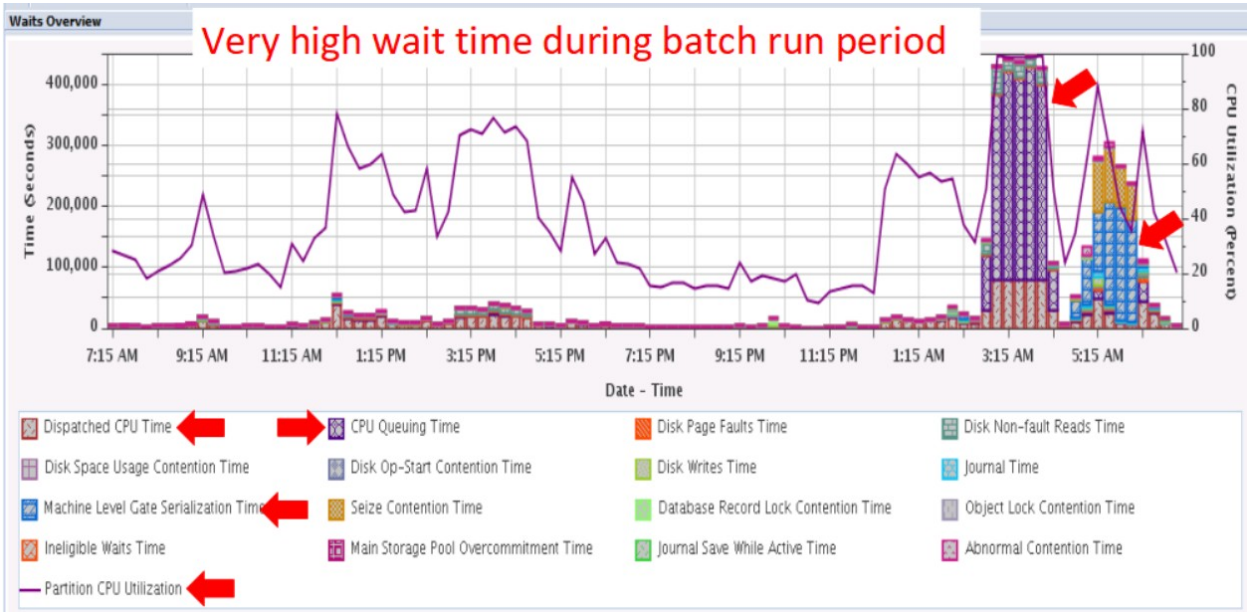

Figure 1: Wait overview chart

In this 24-hour timeline Wait Overview chart above, you can notice that "Partition CPU Utilisation" line graph hits 100% during 3 AM to 4 AM (on the right side of the graph with the upper red arrow). This was the "critical" period of nightly batch process for a customer that I helped with the analysis. You also see that all 5 vertical bar graphs of this batch run period show very large amount of "CPU Queuing Time" right on top of relatively smaller bars of "Dispatched CPU Time" – look at the colour legends on the lower left half of the chart to identify these line and bar graph components. Each individual vertical bar represents a sampling period of performance data – in this case it was set at 15 minutes. So, 5 bars represents a 1.25 hour period.

During this critical period of 100% CPU utilisation, you compare in each vertical bar the proportion of Dispatched CPU Time AGAINST the sum of all wait times on top of it. If the proportion of all wait times overwhelms that of Dispatched CPU Time, it is indicative of a bad performance situation because jobs are kept in wait status much longer than they actively run – and this is the case in those 5 bars above which can also be interpreted as a metaphor that a group of cars (jobs) wait at a red light of one crossroad much longer than they are in motion in this 1.25-hour period.

My personal rule-of-thumb for judging a case of good to acceptable performance is when we see that the PROPORTION of Dispatched CPU Time is NOT less than the sum of all wait times in each bar. This is the case for the rest of the day in the chart above where you can see that the sum of all wait times does not exceed that of CPU time in each bar. (The bars are small but there is "zoom-in" feature for you to use). But if you see overwhelming sum of all wait times in some scattered bars that are not contiguous in time, it may not to be a serious concern – similar to your car waiting at fewer red lights while passing through more green lights along the way to your destination. The excellent performance case is when you see ONLY Dispatched CPU Time without or very little sum of all the wait times in every bar (a sample of this later).

Back to the critical batch period, it should be apparent to you that CPU Queuing time is the only DOMINANT wait component because the sum of all other wait times is very small at the top tip of each bar.

This is the “forest” view of the data as all these times in each bar are gathered from all active jobs during each sampling period. If you see that all bars in the entire 24-hour chart have Dispatched CPU Time as the dominant component with all other wait times barely seen at all, then you can conclude that you have healthy performance for the day. But if this is not the case, you should next drill down to the “trees” view. Since CPU Queuing is the dominant wait in this example, you should next ask yourself whether this particular dominant wait time appears only in one or a small number of jobs or is it distribute amongst most or all jobs?

If you see the latter, it may mean a simple case of overall workload overwhelming the available overall CPU power and you just need to activate more CPU cores to address this situation.

If you see the former that is , the dominant wait time appears in only one job or a specific group of similar jobs, then you have a chance to consider finding ways to modify the jobs in certain manners to reduce the dominant wait or the CPU Queuing in this case. To perform this analysis step, you need to display the chart named Wait for Generic Job or Task as Figure 2.

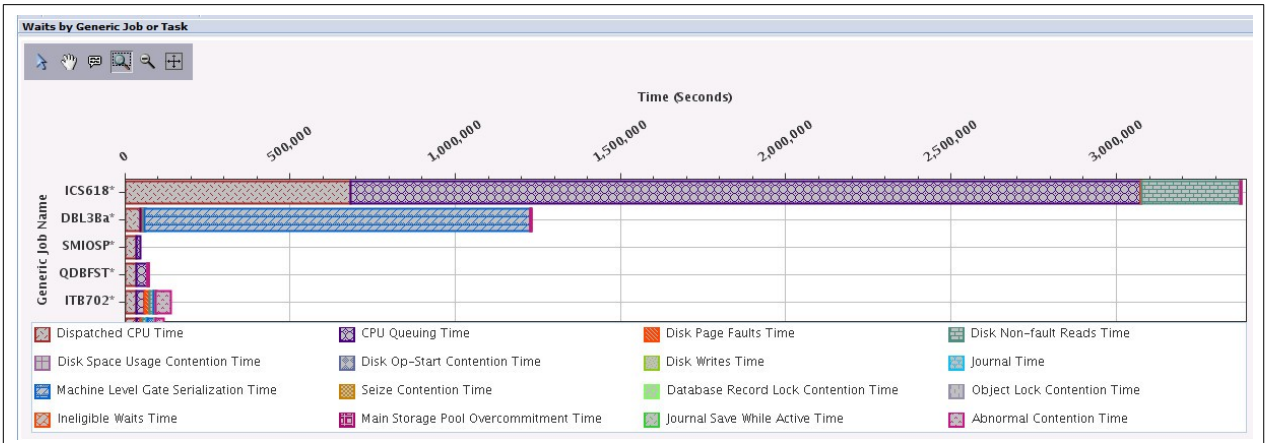

Figure 2: Wait for generic job or task chart

Each horizontal bar in this chart represents a generic group of job names. Each bar is an accumulation of all times in all jobs with the same first 6 characters of their names into one entity. This is based on the assumption that if any jobs running in IBM i share the same first 6 characters of their names, it is very likely that these jobs run the same program codes. They may run concurrently or at different periods in the same 24-hour timeline.

I hope you also notice the generic job names with “*” at the end which means full job names are longer than 6 characters. If you see the names that do not end with “*”, it means full job names are exactly what you see – keep in mind that such a bar may represent one or more jobs. This naming convention is important because you need to use the generic or specific job names you see to check their relevance to what you saw previously in the Wait Overview chart.

Having looked at both charts above, I trust it is clear to you that the generic job named ICS618* (the topmost bar in the chart above) is the only group that accumulates dominant amount of CPU Queuing wait time. I was informed by my customer that ICS618* jobs ran only during the critical batch run period that we saw in the first chart which highlights that these jobs are the main, if not the only, factor causing inordinate amount of CPU Queuing time and therefore deserve to be the focus of our problem resolution consideration. My further inquiry revealed that my customer ran 300 concurrent ICS618* jobs, each of which ran the same group of RPG programs with embedded SQL and/or OPNQRYF.

Let's now try to find a resolution to the issue.

Combining the immense but undesired amount of CPU Queuing wait time with another fact that my customer's server has 10 POWER8 CPU cores, it is reasonable to form an idea that 300 concurrent batch jobs are overwhelmingly too many for the available CPU cores. The customer also told me that this number of concurrent jobs was determined without any specific justification. I figured that it might not be simple to determine the optimal number for such concurrent jobs but since I handled this kind of issue several times before and also happened to be familiar with customer's core application (a popular ISV solution on IBM i), I came up with my personal rule-of-thumb of 5 or 6 “heavyweight jobs” per POWER8, 9, and 10 core. I therefore suggested the customer to try running 60 jobs instead and observe total run-time and wait time result. To everyone's relief, the result was that total run-time of the batch process improved by about 15 minutes from the original 1.25 hours. More importantly, the Wait Overview chart showed that CPU Queuing wait time reduced dramatically to just a small patch on the top of each bar. Further slight reduction of concurrent jobs eliminated all CPU Queuing wait time.

Let me say that it is not batch run-time improvement that is the main benefit here because it is only modest in absolute term. The more important benefit in this case is the elimination of the dominant amount of CPU Queuing time because when this type of wait time appears in an overwhelming amount, it causes any jobs running during that period to suffer severely dismal performance. Now that this wait time is eliminated, all jobs run with decent and consistent performance which all of us love to see.

From my past experience, many IBM i customers (and myself included) struggled with how to determine the optimal number of concurrent batch jobs based on the available CPU power in order to optimise the entire batch run time. Many of them resorted to guessing a high number just to be on the safe side, but the result has frequently been that too much CPU Queuing wait time appeared when I looked at their PDI reports. You need to struggle no more when you utilise the CPU Queuing Wait time analysis as described. And I hope you agree with me that this analysis is quite straightforward once you learn it.

This approach of analysing dominant wait time also applies to any other types of wait time that you see in the legends of the Wait chart. But specific ways to address other wait times are different depending on what they are. For example, if you see substantial Journal Time, you need to buy and install IBM i Journal HA Performance product and turn on Journal Cache parameter in the relevant Journal object(s). If you see substantial Disk Page Fault Wait time, you either have bad disk response time and/or abnormally high memory page fault or both and have to upgrade disk HW and/or identify jobs that produce abnormally high memory faulting rate and try to reduce it. (I may talk about this in another occasion.) The list goes on.

You may now wonder how a good overall system performance looks like in Wait Overview chart? I have an example in Figure 3.

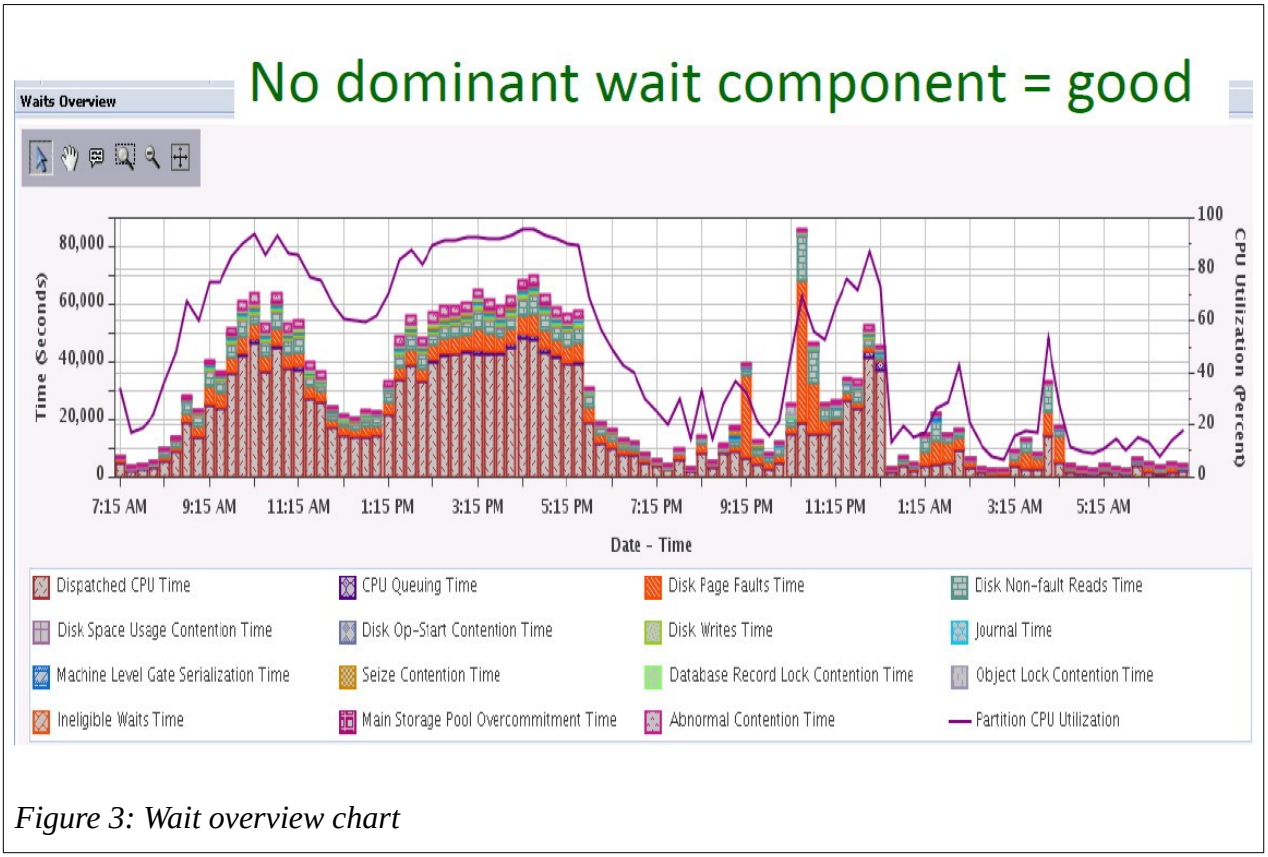

Figure 3: Wait overview chart

You see here that in each of all vertical bars of this 24-hour chart, it is the Dispatched CPU Time that is the persistent overwhelming dominant component against the sum of all wait times appearing at the small tip of each bar (with few exceptions which means no worry here). In such a case, the overall CPU utilisation in the entire daytime period is consistently on the high side but you can hardly see CPU Queuing wait time at all.

This last chart is a case of a very decent overall system performance from IBM i Server hardware and some OS perspectives. I t does not always mean there is no performance issue if your IT infrastructure environment involves multiple servers working together which is quite typical of contemporary client/server and web-based application deployment. If any such application users complain about bad performance of their business operations BUT you see the Wait Overview chart that exhibits little overall wait times like the sample above, then it means that the cause of the problem is not in IBM i server hardware nor most OS functions. The rarity of overall wait times speak for themselves. We need to look elsewhere for the cause of performance issue. I used to help some customers with such an issue of external causes of performance problems and I intend to share with you on this in another article.

The last point I would like to mention here is that, in the past, IBM used to publish guideline on CPU Utilisation % for many old AS/400 and iSeries models. But as of the era of POWER5-based servers when IBM delivered virtualisation technology such as virtual processor, uncapped partitioning, shared processor pool and such, CPU % Busy guideline became less useful. You may see your server's CPU frequently utilised at 90-100% all day long but as long as you see little or no overall CPU Queuing (and Machine Level Gate Serialisation Time. but I do not discussed this here), there is no immediate and serious cause of concern about bad overall system performance in IBM i server. But you still have a long-term concern on either server's CPU power sizing or deployment optimisation of your application workload or both.

I do hope you now gain some understanding and are appreciative of these two useful Wait charts of the PDI tool and I ask that you will analyse them in the future for the sake of your deserved benefit.

IBM i performance investigative game is afoot!

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.