IBM i: Object authority check and batch job run-time

Satid Singkorapoom

Batch processes are common in nearly all kinds of business applications and time to time, many customers have to deal with batch runs that take too long. Solving these problems may not be easy as there are often many factors that influence run times. One such factor in IBM i is how you assign object authority access rights for user profile(s) running the jobs to the libraries and their objects being accessed by batch jobs. The following case study demonstrates how important the correct assigning of access writes can be.

> Note: I assume that readers have basic knowledge on how object access rights are assigned in IBM i.

Many years ago, I was asked to solve a problem for a customer with a batch run time that was unacceptable. The customer ran 25 concurrent batch jobs in the batch process every night (on a POWER7 server with 7 CPU cores) and the entire process took about 8 hours to finish and was causing serious concern for the customer. They wanted the run time to be substantially shorter and my local IBM SW services team was contacted to help. My services team colleague collected all required performance data and sent it to our WW IBM i support team and the WW team eventually provided a feedback that all performance factors in POWER7 server HW and OS looked good (meaning not being utilised beyond their capacity) with the exception of a quite high “object authority check” overhead (this performance data is in IBM i performance database table named QAPMSYSTEM[1]. At the time, no one knew what could have been done to reduce this overhead and this was when I was asked to step in to provide a solution to the issue based on the feedback from the IBM WW team.

I remembered that I had produced a performance report and had also noted that the server HW was quite powerful and generally under utilised during the batch run as was originally observed by our WW support team. But this was the first time I ever encountered this object authority check overhead factor and its effect on batch run-time.

Luckily in the early days of my IBM career, I delivered on a number of occasions IBM i customer training for a standard class named “AS/400 Security Concept and Planning”. I quickly recalled the diagram (Figure 1) that explained the IBM i search order for a resolving object access rights of a job's user profile to an object being accessed by the job. The search stops at the step where it first determines a clear access right that either allows or rejects the manner of access (read, update, add, delete, run, etc.).

1 https://www.ibm.com/docs/en/i/7.1?topic=data-collection-services-files-qapmsystem

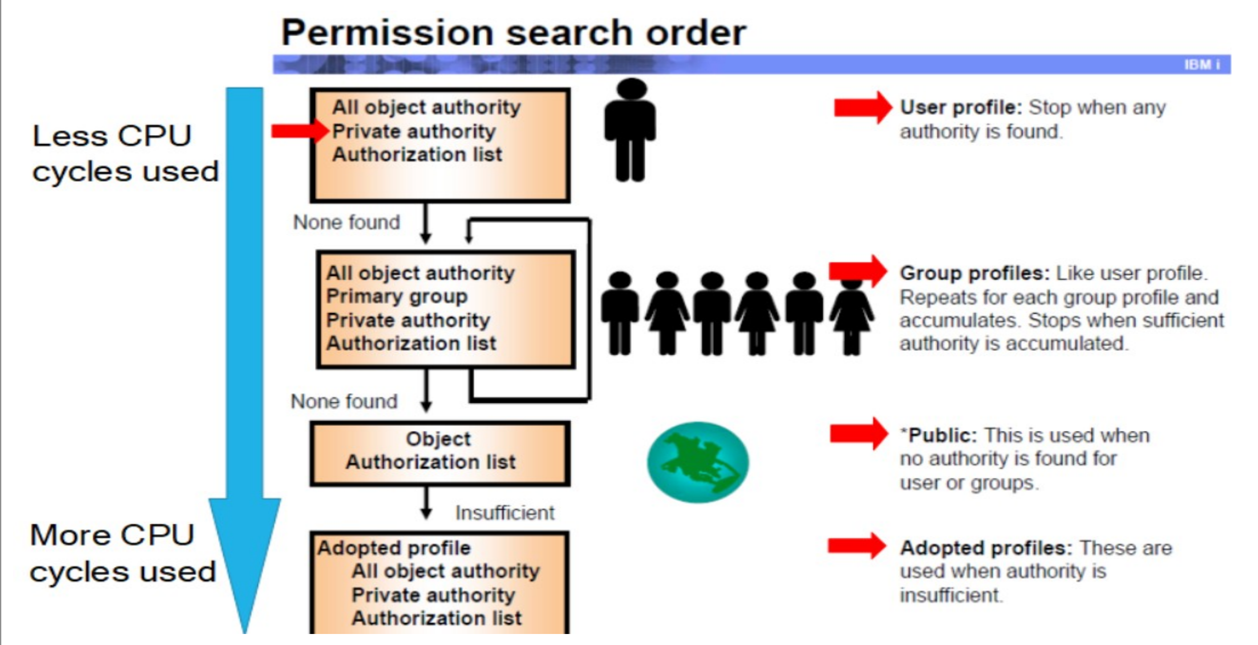


Figure 1: Permission search order

Briefly the Figure 1 shows that IBM i determines whether or not an object access rights assignment can be found in the user profile that runs the job first, starting at whether there is *ALLOBJ special authority in that profile or not. If not, it proceeds on to the next step. If no concrete access right assignment is found (allowed or rejected) in user profile (by private authority or next by authorisation list), IBM i proceeds next to check for it in the group profile(s) involved (which is actually another user profile object) and then public authority of the object being accessed and lastly in the adopted user profile, but only if "adopted authority" is used. The deeper IBM i has to traverse down the path in the diagram, the greater the processing overhead is for the system task that executes this operation of the batch job.

But if a job accesses an object only once, then even if IBM i has to traverse the entire path to resolve to a concrete access right, this does not waste a lot of CPU cycles at all. For example, if I run a program that adopts authority of another user profile just to retrieve a record from a physical file and no more, then IBM i traverses almost to the bottom of the search path above but this happens only once and there is little overhead. However in a typical business database processing environment, it is normal that many concurrent jobs are running and each job accesses data records often repeatedly from many physical files (open file – access data - close file – open again and so on). This will invoke a large number of authority check traverses and accumulate a substantial overhead. This is exactly what my customer was experiencing.

With all these understanding in mind, I checked with the customer how the user profile, that ran the 25 batch jobs of concern, was assigned the object access authority to the libraries and their objects of the core application. I was informed that all 25 jobs ran under one user profile for each day and access authority was granted by way of a group profile (to which the user profile belonged). This was specified in an authorisation list with proper access right assigned. Consulting the search path diagram above, you can now see that it takes at least 7 steps down the entries in the top two rectangles (first for user profile and the next for group profile). I say "at least" 7 steps because if the check in the first group profile of the user (a user profile can be assigned to up to 16 group profiles) does not yield the access right determination, the next group profile is checked (and thus the "loop" indication for the second rectangle in the diagram) and more overhead is incurred in object authority check process.

When I asked the customer, they told me that they used group profile in an authorisation list to emulate what was implemented in the core application for “interactive” user profiles. They were not aware of this object authority check overhead before.

At this point, I hope we can now see that to reduce this overhead, the authority search path diagram tells us that we should assign EITHER *ALLOBJ special authority for the user profile that run the batch jobs (which incurs the least amount of the overhead) OR assign the user profile with proper access rights directly to all application libraries and objects within (“Private authority” entry in the diagram) that the batch jobs access. Using *ALLOBJ is normally NOT allowed in many companies as it violates prevailing IT security policy, but I will discuss later about how you may use this method without the violation.

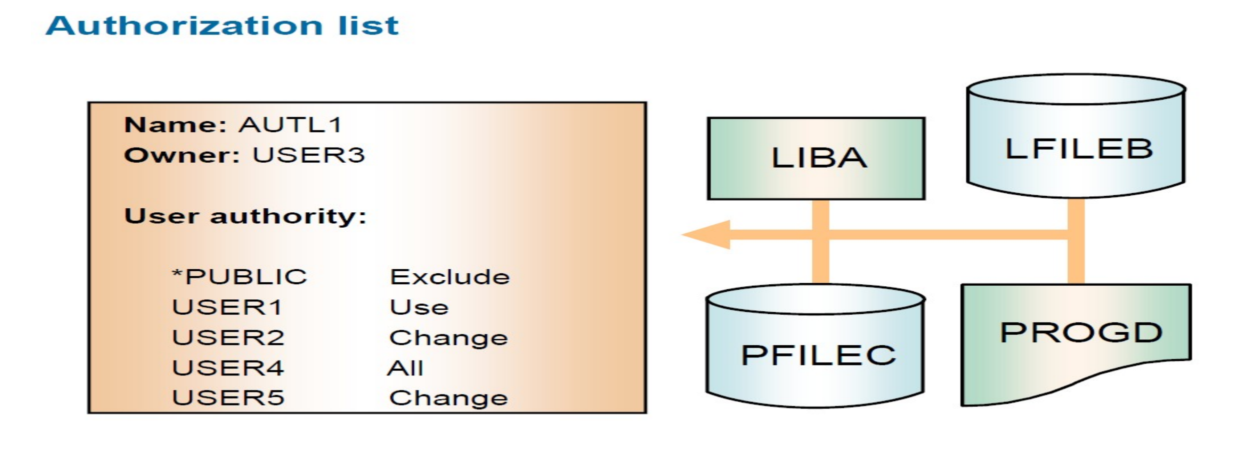
I informed the customer accordingly and told them that they could determine the best case scenario of the run time improvement by first trying *ALLOBJ and thereafter changing to private authority to compare the result with the previous one and deciding on whichever method they see fit for their case. I was later informed by the customer that the trial run of *ALLOBJ yielded the run time of about 4.5 hours (down from 8 hours). This was quite a dramatic revelation for all involved including myself. All of us recognised the importance of this factor right away and I never forgot it ever since.

After this case, I have been involved in similar problem resolution support for many more customers and I carry this knowledge with me always.

Having learned about this case study, you may wonder why we need to use group profile and authorisation list then? Let's see why.

When business applications are not of GUI-based, client-server type, all application users run their own non-batch jobs in their own telnet sessions and there can be hundred, if not thousand, of such users active in an IBM i server everyday. Assigning *ALLOBJ or private authority is impractical, even impossible, to implement. Therefore, group profiles and authorisation list are introduced to help system/application admin deploy less complicated and therefore efficient object access rights control to all “interactive” users in the server.

Group profile helps you treat a number of user profiles commonly like one in terms of their data access rights. Authorisation list helps you treat a number of objects under the same set of data access rights assignments as demonstrated in a Figure 2 below.



Name: AUTL1
Owner: USER3

User authority:

*PUBLIC	Exclude
USER1	Use
USER2	Change
USER4	All
USER5	Change

Figure 2: Authorisation list

As an example of the benefit you can get, in a server with a hundred interactive application user profiles, it is possible to deploy, say, 30 group profiles and 15 authorisation lists to assign access rights for all these users to all application objects. This results in much less security-related entities to mange. This concept exists in virtually all major server operating systems because it helps simplify system administration of these users of non-batch jobs.

But you may ask what about the object authority check overhead with group profiles and authorisation lists for these interactive users? When it comes to interactive jobs as opposed to batch jobs, the former tend to be a series of relatively short run-time sessions running programs that access less amount of data from less number of application objects than batch job which means object authority check overhead can therefore be relatively low.

On the contrary, a typical customer would need only a few user profiles to run their batch jobs that usually access much more data from many more application objects. So, there is less need to use group profile nor authorisation list in this case.

I hope you now appreciate that using different object authority assignment approach for interactive job users to batch job users would help you ensure minimum run time overhead for the latter.

One last note on *ALLOBJ. I mentioned earlier it typically violates IT security policy to have more than one or two user profiles in your system with this special authority. But if you are in dire need of *ALLOBJ because the best run time of your batch process is of critical to your business operation, what could you do that will not violate the security policy?   I have an idea to share for your consideration.

Basically, you should create a user profile with *ALLOBJ and ensure this profile cannot be used to log on to your system and no one can refer to use this profile from SBMJOB command nor profile swap API. This profile is used exclusively to run the only the batch jobs you need. Then you specify this profile to run all your important batch jobs from a set of “launching” programs that use adopted authority.

First create a user profile with special authority (SPCAUT) = *ALLOBJ AND also with the following profile parameters: STATUS = *DISABLED, PASSWORD = *NONE, INLPGM = *NONE, and INLMNU = *SIGNOFF so that no one can access your system with this profile – let's call this profile SUPERMAN. (A disabled profile cannot run interactive jobs but it can run batch jobs.)

Then use EDTOBJAUT command to assign “exclusionary” access right to this SUPERMAN user profile object so that its PUBLIC authority is set to *EXCLUDE and remove all its private authority entries. Also change the owner of SUPERMAN to QSECOFR. (No need for this if you use QSECOFR to create SUERMAN.) This prevents anyone (except for user with *ALLOBJ) to use SBMJOB command or profile swap API and specify SUPERMAN to run any other jobs.

Then you identify all the program objects that launch all batch jobs (or compile new programs) and change the owner of these programs to SUPERMAN. Also change program object's parameter USRPRF to *OWNER (default value is *USER). If these programs use SBMJOB command to launch batch jobs, change the USER parameter of SBMJOB to SUPERMAN. If the programs use profile swap API, specify SUPERMAN as the user name to swap to.

Lastly when the batch jobs run, use WRKACTJOB to ensure that they run under SUPERMAN. If not, you may need to check the job log to find out what has gone wrong.

However there will be some customers that are unable to modify their code as recommended above, but I hope that they still understand the logic and can adopt it to their particular circumstances.

Now, I hope you clearly understand the different ways to correctly deploy IBM i object access right assignment between user profiles of interactive and batch jobs in order to optimise the performance of the latter.   I'm confident this piece of knowledge will pay its dividend handsomely for all of you who value the performance of your batch processing.

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.