What I Learned This Month: Server Address Spaces and SYSSTC

Scott Chapman
American Electric Power

This month I'm back to discussing z/OS since I had an issue with Workload Manager (WLM). As a reminder for those who don't deal with WLM every day, z/OS manages work based on "service classes". All work in the system is assigned a service class based on the WLM policy. There are two system-supplied service classes that have fixed dispatching priorities. SYSTEM is the highest priority (255) and is generally used for operating system address spaces. SYSSTC runs at the next highest priority (254) and generally is used for highly important started tasks that are processing application work. It may also be used for somewhat less important started tasks that are well behaved and consume relatively little CPU. WLM automatically manages the priorities of the other services classes based on their goals and importance levels.

I've been chasing a strange problem with some of my WebSphere (WAS) servant[1] regions. Periodically (at both quiet and busy times) there were unusually long delays between the request arriving at the controller and starting in the servant. Looking at RMF, I noticed that the servant appeared to be logically swapped out, which certainly explains why the work wasn't dispatched. What I couldn't figure out is why the address space would be swapped out.

As luck would have it, Peter Enrico was in town for the Ohio Valley CMG meeting and he graciously agreed to look at the problem data. He also found my problem puzzling, but another thing that struck him as odd (I think he might have used the word "wrong") was my choice to put the servants in SYSSTC.

The WAS servants are WLM-managed "server" address spaces: they execute application transactions. WLM manages such servers to the goal of the transactions. But not everything that happens in the address space is necessarily directly related to a transaction. For example, server startup or certain background JVM housekeeping activities aren't related to particular transactions. Measurements for such work are accumulated in the original service class: the one that the address space was originally classified as. In the case of my production WAS servants, that was SYSSTC. However, once transactions start arriving, WLM manages the address space based on the performance of the transactions and will not manage it as SYSSTC.[2]

---

[1] The servants are the address spaces that execute the application transactions. There are various other WAS address spaces that get used for functions other than actual application execution.

[2] To be only slightly more specific, the address space gets assigned to an internally created service class period. Somebody (not me) could write a much longer article on all those details that are mostly internal to WLM and not very important to this discussion.

And this is the crux of Peter's argument: if server address spaces are assigned to SYSSTC, any reporting on the utilization of SYSSTC will include the time the server address spaces are running at a lower importance (and lower dispatching priority) than SYSSTC. This makes the SYSSTC numbers misleading because most people would look at the SYSSTC measurements and presume that all the CPU time was consumed at dispatching priority 254. That's not the case if you have your server address spaces in SYSSTC.

Also, if you've put the workload in SYSSTC to avoid using another server class period[3], you haven't really succeeded since WLM creates special internal service class periods for servers it manages.

Peter also pointed out that SYSTEM and SYSSTC are ineligible for proactive storage protection, and won't be managed for storage if paging occurs. My systems generally don't page much at all, but that's a good point to keep in mind.

In my particular case, I put the servants in SYSSTC some time ago to help during startup processing of the servants and because the servants execute mostly on the zAAPs, not the GCPs[4]. But when they need to get on the GCP, I really do want them to get access to it.

After some discussion, I came to agree with Peter that it would be best to keep server address spaces out of SYSSTC as that does reduce the potential for confusion in interpreting the CPU utilization of SYSSTC. I have been meaning to adjust my importance levels across the board and this is a good excuse to do it. There's no reason why I can't satisfy the WAS servant's needs with an importance 1 service class instead of SYSSTC. I may have to move some things down to my under-utilized importance level 2, but that would help spread work more evenly across the importance levels, which is also a good thing.

Just to be clear, absent any storage (paging) issues, I don't believe that having server address spaces in SYSSTC is necessarily a performance problem. However, it may create confusion that might make it harder to diagnose a performance (or capacity) problem. At least I don't believe it should be a problem. However, the initial response from IBM support regarding my original problem is that they believe that WLM is making an improper adjustment to the number of desired active tasks when the servant is classified to SYSSTC and MANAGENONENCLAVEWORK=YES is set.

Using MANAGENONENCLAVEWORK when you have servants in SYSSTC also struck IBM support as particularly "unusual". I did this intentionally because we were severely CPU constrained and at times it appeared that the servants were waiting an inordinate amount of time to get dispatched on the GCP to do a tiny

[3] Remember that you should not have more than about 25-30 active service class periods at any one point in time on any given system.

[4] There are multiple types of CPUs on a mainframe: General Purpose CPUs can execute any work. I call them GCPs for short. Java work can run on CPUs that IBM calls zAAPs (or sometimes processors called zIIPs). All the CPUs are physically exactly the same, and a discussion of why such different logical processor types exist is beyond the scope of this article.

amount of work. The combination did help reduce the GCP CPU delay without a significant impact on the other workloads because the majority of the servant work is actually done on the zAAPs.

At the time I'm writing this, IBM is still evaluating my original issue of the servants getting swapped out. I did find though that moving the servants out of SYSSTC seems to prevent them from being swapped out. I don't know if that's working as designed or is a bug, but apparently following Peter's advice does prevent the problem.

So that's now three potential reasons to avoid putting server address spaces in SYSSTC: it makes the measurements of SYSSTC misleading, such address spaces won't get proactive storage protection, and in some cases it might be causing WLM to make an improper decision. Although I suspect the last point is really a bug that IBM will eventually fix.

I also think this is another example of how it is usually better to demote workloads to lower importance levels instead of arbitrarily raising suffering workloads to higher importance levels. If I had done that instead of raising my servants from importance 1 to SYSSTC, then I wouldn't have had this issue and my SYSSTC numbers wouldn't be misleading.

Finally, it's always valuable to get to spend some time with somebody who understands things at a deeper level than yourself. Always avail yourself of such an opportunity!

I hope you found something interesting and useful in these tips. As always, if you have questions or comments, you can reach me via email at sachapman@aep.com.