What I Learned This Month: Spring Cleaning of the WLM Policy

Scott Chapman
American Electric Power

As I write this, spring is trying to come to my part of the world—the flowers are up and I clearly need to get my lawn mower blades sharpened. (So I can mow the grass, not the flowers!) Some have a tradition of "spring cleaning" their homes. Perhaps we should also have a tradition of "spring cleaning" our WLM policies.

Certainly, if you are a mainframe performance specialist that has been paying attention at CMG over the years, you know that there are several best practices for your WLM policy: use reasonably attainable goals, keep things as simple as possible (avoid excessive active service class periods), have some work classified as discretionary, and periodically review things to make sure that everything is still reasonable and correct. If you're like me, though, you probably don't regularly review your policy.

While addressing an issue recently, I took the opportunity to do a little spring cleaning of my WLM policy. Here are a few things that I found; perhaps the list will inspire you to look at your policy.

I had a few obsolete service classes, report classes, and resource groups that were no longer being used. When I stop using these definitions, I generally don't delete them immediately on the off chance that I want to put them back. Not that I couldn't restore the policy from a backup, or just recreate them, but it just feels better to leave them defined for a little while. Of course "a little while" tends to end up being years. Oops. Such unused constructs don't cause any overhead that I'm aware of, other than visual clutter that might lead to future confusion. So I expunged the things that I should have gotten rid of some time ago. I've decided that going forward, since I always document my changes in the "notes" section of the policy, I'm going to add a line at the bottom of the notes (where I'd put the next entry) to delete the obsolete construct upon the next policy update. Hopefully that will help avoid future confusion for myself or my successor.

We let most production batch work go to one service class, but have specific overrides for certain jobs to direct them to our PRDBATHI service class. I found that we had a number of jobs going there that made sense 5-10 years ago but did not make sense today due primarily to hardware changes that improved performance across the batch window a couple of years ago. There simply isn't as much of a need to try to boost particular critical path jobs. Plus the usual critical path has changed over the years as well. I ended up removing 90% of the workloads that were being directed to PRDBATHI. That resulted in a small overall improvement in the batch window performance.

Velocity goals should be reviewed at least whenever you change processor configurations, and I do that. But even better is to review them on a regular basis, and I'm not nearly as good at that. Reviewing our batch velocities resulted in a minor change to the goals for production batch. It doesn't appear that the change was material, but it was perhaps a small step in the right direction.

Resource groups also need to be reviewed if you're still specifying them in service units. When I reviewed them, I decided to switch all but one resource group from that old method to percentage of the LPAR share. That simply makes more sense to me, and is probably a better, more dynamic rule. Instead of saying that something can only use x SUs, I've said that it can only use y% of the LPAR. If the LPAR capacity changes, the resource allotments then stay the same relative to the other workloads on the system. Of course, I can see situations where that wouldn't make sense, but for most of the situations where I use resource groups, I think it makes a lot of sense. Letting WLM calculate the exact SUs required is easier on my brain.

You may be wondering what started me on my WLM spring cleaning adventure. We were having some issues with some of our DDF work, which is generally classified at a similar importance level to batch. That classification was causing trouble for certain short-running DDF work during the batch window. Due to the issues we've seen in the past with unexpected large increases in DDF workloads, often due to application problems, I'm very reluctant to assign too high of an importance to it. I took a look at the normal utilization of the period 1 work during a non-problem time and saw that it was on the order of less than 1% of the problem LPAR. So I gave it a minimum resource group of 1% under the theory that if something started running amok, 1% is not an onerous amount of capacity to give up to it yet 1% should be sufficient to get the short-running work done. That didn't completely eliminate the interference from the batch window on the DDF period 1 response time, but it greatly reduced it and allowed DDF period 1 to meet its goal. And because I didn't change the relative importance levels, I haven't really increased the risk of a surge in DDF work causing problems for other workloads.

So what did I learn this month? First, spring cleaning should probably apply to our WLM policies as well as our homes. Second, minimum resource groups can be useful for protecting short-running work that's in period 1 of the service class in question.

As always, if you think I didn't learn my lesson correctly, please email me at sachapman@aep.com and let me know!