What I Learned This Month: End-of-Year Collection

Scott Chapman
American Electric Power

This is the last issue of MeasureIT for the year, so I thought I'd wrap up 2011 by collecting a few small things that didn't make it into a "What I Learned This Month". Hopefully you will find something interesting in this collection!

Probably most mainframe professionals know exactly what an S322 abend is: CPU time limit exceeded. So when is an S322 not an S322? When it's an SEC6! I ran into this when a multi-threaded application ran out of CPU time while one of the TCBs was in a system sleep call. Apparently when a Unix thread wakes up to find that it has run out of CPU time, the result is an SEC6. An EC6 by itself is a catch-all: "An error occurred during processing a callable service". I don't remember the exact reason code we got, but it wasn't anything that clearly indicated "out of CPU time". Also, when the CPU time limit is reached in that manner, the address space is allowed to run for some time to allow the sleeping threads to wake up and clean up. So the application wasn't ending at quite exactly the same amount of CPU time every time, although it was a strangely similar result each time, which should have been our first hint.

The other thing I discovered during that problem is that my shop has a different CPU time limit for started tasks than batch jobs. I had always assumed that the started tasks had the same hour default that we give batch jobs, but it turns out that they only have 15 minutes by default. Almost all of the started tasks run with a specific TIME=1440 or NOLIIMIT, which is probably why it never occurred to me that the default might be different. The reason why our default for started tasks is less than batch jobs is apparently lost in antiquity. I had hoped that I had learned not to assume things by now, but that appears to be a lesson I have to periodically re-learn.

Speaking of CPU time limits, specialty engine (zIIP and zAAP) CPU time counts towards your job limit and the amount accumulated is normalized to the speed of the GCPs (general purpose CPUs). This all makes sense, but I hadn't really thought about it until I saw a Java job take an S322. I suspect that I could override this in the IEFUSI system exit, but I haven't looked into that yet, nor have I decided whether I really want to eliminate the specialty engine CPU time from consideration for the CPU limits.

And speaking of zIIP and zAAP usage, time a workload spends on zIIPs and zAAPs doesn't count toward any resource group that might apply to the workload. If you have a Java batch job that's effectively running 100% on a zAAP, you can't limit its usage of the zAAP by assigning it to a resource group that has a usage cap. That zAAP and zIIP time doesn't count to the same resource group minimums and maximums makes some sense, but I can foresee a situation where a shop with lots of Java work might like to have all the controls for those Java workloads that they have for their traditional workloads. In that case, it seems that it would be nice to have resource group controls for each type

of processor, just as PR/SM manages each type of processor separately. I suspect IBM is not in any hurry to add that complexity to WLM.

One answer might simply be to buy more zAAPs--they're "cheap". Maybe, but there still are limits as to the number of specialty engines you can buy, so that answer has its limitations. And I might argue that while zAAPs are "cheaper" than GCPs, they're certainly expensive compared to CPUs on smaller platforms. If you can't use all of z/OS' great management functions to control your Java workloads (although you can use most of them), there is perhaps one less reason for deploying your Java work to z/OS. But, this is likely a problem only for sites that have a significant amount of Java work on z/OS and have specific needs for managing those workloads.

Finally, since we're talking about how much things cost, the mainframe performance analyst and capacity planner's job has gotten more complicated over the past several years. We still need to understand our business work, the trends for that business work, and the relationship of the business work to the technical capacity measures. However, we now have three separate CPU pools[1] to be concerned with: GCP, zIIP, zAAP. We need to understand how and when work gets routed to each of those processor types. The superscalar CPUs in the latest generations of mainframes make CPU measurements more variable than they used to be and CPU consumption per unit of business work doesn't necessarily remain as fixed as overall utilization increases. I've seen normal CPU variability for "traditional" workloads on the order of 5-10% on our z10 machines and I've seen variability for Java workloads of 20-70%! I'm still working to understand the Java variability I've seen.[2]

Part of today's business challenge is to "do more while spending less", so today's capacity planners need to understand costs, too. Specialty engines have different cost profiles than the GCPs, which could be very important if you're truly trying to think outside the box to save money. Migrating a process to a less efficient Java-based solution might be a good decision if it means offloading workload from the GCPs to zAAPs. Since the software often (always?) costs more than the hardware, capacity planners would do well to understand how their mainframe software is licensed and how the costs are calculated. Unfortunately, the software licensing often seems even more complicated than the technical challenges mentioned above.

I guess the good news is that we'll all have ample opportunity to learn more things in 2012! I hope to continue to share what I learn with you in MeasureIT next year.

As always, if you think I've got it all wrong, please email me at sachapman@aep.com and let me know!

---

[1] Yes, I know about zAAP on zIIP, but that's not a good fit for every site. And of course we can't forget about ICFs and IFLs as well, but I'm speaking of CPUs that z/OS applications run on.

[2] Shameless plug: I'll be talking about some of my Java performance experiences at CMG'11 in December in Washington DC. I hope to see you there!