What I Learned This Month: PDSE Buffering

Scott Chapman

American Electric Power

There's always a tension between exploiting new features and functions immediately and adopting the "if it ain't broke, don't fix it" philosophy. In the past, when we had more people, more time, and less restrictive change management practices, it was easier to experiment with new features. Today, that's not the case in many organizations.

So it's not terribly surprising to me when we have a problem that can be solved by enabling a new feature or function. This time it was PDSE[1] tuning. While PDSEs have been around for a long time, adoption in our shop has only taken off significantly in the past couple of years.

In this particular instance, I came into work as usual at 6:30 AM and noticed that we had some critical overnight batch jobs still running—about 8 hours after they are normally done. I found a PDSE that seemed to be receiving over 1,000 I/Os per second. That's an unusually large amount of I/O for a single dataset to sustain for hours. Response time wasn't a problem at 0.3ms; clearly, the I/Os were all cache hits.

This particular PDSE was the Fault Analyzer history dataset. Fault Analyzer is an IBM product that captures diagnostic information whenever the application takes a dump. In this particular case, the application architecture is such that when a called module finds an error condition, it takes a dump then returns an error status to the calling module. In this situation, the calling module ignored the raised error condition and was continuing processing with the next account.

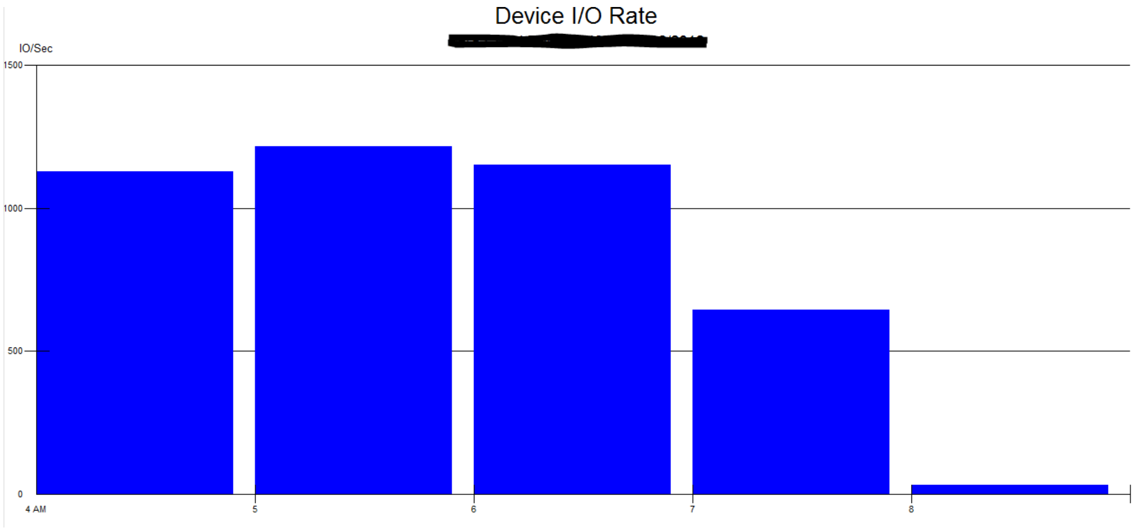
Taking a dump is an expensive process. Doing it repeatedly will hurt performance. We designed the application architecture in the early 90s with the idea that when a called module returned an error code, the calling module would bubble the error up until the job ended. The application programmer would get called to correct the error situation. Alas, we underestimated the ability for the application code to generate errors. Over the years, there have been various incidents similar to this where the calling module would ignore the error condition and generate dozens of dumps out of a single job. But none were as severe as this—the job was dumping literally hundreds of times.

While the application team was looking into the error condition itself, I set to looking for a way to reduce the I/Os during the dump. If you're thinking "those I/Os are cached, so why worry about it?", you didn't do the math. Over 1,000

[1] For the people who aren't mainframe literate, a PDS is a "Partitioned Data Set". It contains "member" files within a single dataset. In very loose terms, you could think of the PDS as a zip file and members as files within that zip file, except that there's no compression involved. Probably close to 20 years ago, IBM introduced PDSE (Partitioned Data Set Extended) to solve some of the limitations in the original PDS structure.

I/Os per second times 0.3ms each means over 30% of each second is spent doing I/O. Eliminating I/O wouldn't eliminate all the overhead of taking the dump, but it would help.

Not surprisingly, there were a number of PDSE tuning options that I'd never implemented. The first one I noticed that seemed to apply was "buffer beyond close", which was an option introduced a few years ago that retains PDSE buffers for both directory and member data after the PDSE is closed. That seemed perfect since the dump history file was being repeatedly opened and closed to take the successive dumps. Fortunately, that option can be dynamically enabled and I did so around 7:30 AM. You can see the impact on the I/O in the graph below. By 8:00 AM, (the hour at the far right of the graph below) the I/O/sec rate had decreased to a negligible amount.



Later in the day, we also reduced the size of the history file from about 50GB to about 8GB. The 50GB size had been chosen early in the implementation of Fault Analyzer before we convinced the application team to suppress duplicate dumps. With the duplicates suppressed, 50GB contained months of dumps, which was unnecessary. The smaller file contains a couple of weeks' worth of dumps and has a much smaller directory to traverse, further improving performance for the following evening's batch run.

A couple of days later, the application team also determined that the "error" condition really wasn't an error and so stopped treating it as such which eliminated the dumps all together.

There were multiple lessons learned here:

- The "Buffer beyond close" option can be very helpful for certain active PDSEs.
- The old adage "the only good I/O is no I/O" still holds true, even when you're getting 0.3ms response time.
- Reasonably sized PDSEs perform better than larger PDSEs.
- Don't repeatedly generate dumps out of the same batch job!

- Sometimes it is worthwhile to proactively fix (e.g. implement buffer beyond close) what’s not broken, before it breaks.

As always, if you have questions or comments, you can reach me via email at ~~sachapman@aep.com~~ scott.chapman@epstrategies.com.