What I Learned This Month: Data Compressibility Matters

Scott Chapman
American Electric Power

Recently, for the first time in a very long time, I had a tape performance issue to investigate. That's not because we're running such great new hardware: our peer-to-peer B20 VTS certainly shouldn't be expected to keep up with more modern hardware, but it generally performs well enough for us. But in this particular situation, the DB2 log offloads fell too far behind and DB2 had to stop doing work to wait for the offloads to catch up. The obvious "real" problem here was that the DB2 logs were not sized for the level of activity that was being generated.

However, in looking at the offload times, it appeared that the offload throughput was in the single-digit MB/s range. That seemed a bit too slow, so my first thought was "how busy is the VTS?" because certainly the VTS has an upper bound for its rated write throughput. I have tracking in place for those sorts of statistics and that value didn't look like a problem: total write rate at the time was less than 100MB/s and the rated value is somewhere around 180MB/s.

So I checked that off as being OK and looked for bottlenecks elsewhere. Unfortunately, I couldn't find any: channels, CPU and DASD all looked perfectly fine. Looking at the history of the DB2 log offloads I found that they regularly underperformed at times where the VTS write throughput was at similar levels. That led me back to looking at the VTS. One thing I noticed was that during the problem intervals, the hardware was reporting a compression ratio much lower than our historical expectations: as low as 1.2 to 1 at times. That matters because the write throughput that the VTS is capable of achieving is based on the host write rate (e.g. 100MB/s), the block size, and the compression ratio. The throughput ratings are based on assumptions about the compression ratio—probably something on the order of 2.7 to 1, not 1.2 to 1.

For years I've been tracking the host write rate, but the better thing to track seems to be the actual back-end (after compression) rate. Going back and calculating that out over an extended period of time revealed that we rarely were driving a back-end rate above 50-55MB/s. When we did push into that range, the host write throughput suffered—there was a noticeable increase in elapsed time for the log offloads that always write the same amount of data. I concluded that was our real limit. There is some variability in that, presumably at least in part due to the block sizes used at the various times.

To validate that host throughput suffered at these busy times, I set up test jobs that copied a specific amount of data to tape every hour. One of those datasets contained data that was already compressed and hence there was little opportunity for further compression. Another dataset contained more standard textual business data that

should compress fairly well. Both datasets were the same record length and block size and contained the same number of bytes. The same datasets were written during each test. As expected, when the back-end rate reached into the 50MB/s range, the time it took to write the data to tape increased significantly. And in general, the already-compressed data did take longer to write than the compression-friendly data, particularly at peak times. This makes perfect sense when you think about it. At a given host write rate, the non-compressible data will consume more of the available back-end bandwidth and so it should be more sensitive to the availability of that resource.

Out of this exercise, a few other interesting points were uncovered. One is that the capabilities of our CPU and DASD have far outstripped the capabilities of our much older tape infrastructure. That unbalance leads to tape performance bottlenecks at peak times. Normally we don't care that the backups are taking longer, but it does mean that critical tape users (such as the DB2 log offloads) are at risk.

Another interesting discovery was our overall compression ratio. I hadn't calculated that for quite some time; the last time I had it was around 2.5 to 1. Now it's running closer to 1.8 to 1. As we look to replace the VTS, understanding that compression ratio is critical, and assuming that we were still running close to 2.5 to 1 was a bad assumption. The reason for that change is relatively simple: most of our recent growth has been in DB2 and most of our DB2 data is compressed. Also, CICS is now compressing its SMF records and we write a fair number of those records every day. So it makes sense that the compression ratio would not be as good as it used to be.

The final thing I found interesting was that I expected that moving from a relatively poor 24KB block size to 256KB blocks would provide a noticeable improvement to my test jobs. But it didn't seem to make any statistically significant difference, whether the VTS throughput was constrained or not. I don't know exactly why that was; it's just what I saw. But it is another example of a long-held belief (bigger blocks are always better) that may not be as meaningful as I once thought it was.

So that's what I think I learned: our VTS capacity is dependent on the compressibility of the data and our data has become less compressible over the past few years. And yet again, existing "rules of thumb" and assumptions need to be periodically revisited. Seems like I was supposed to learn that one some time ago. You might wonder if I opened a ticket with the hardware vendor, and the short answer is that I did not because we expect to replace the VTS soon and it's not causing a regular problem.

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