What I Learned This Month: DB2 Compression: Still Good

Scott Chapman
American Electric Power

Every now and then one should revisit long-standing decisions to see if they still make sense. Of course it's easy to say that and it's easy to agree with that sentiment, but it's harder to actually do it. After all, there are so many decisions that have been made over time, it's hard to determine an appropriate review interval for each and every one. One answer is that as you get more desperate to find "opportunities", you should get more aggressive about revisiting prior decisions.

In the mainframe world (at least, but I believe not exclusively), the real cost of capacity is not in the hardware cost but in the software cost. Unfortunately, software costs sometimes make us contemplate decisions which are technically inferior, but which may reduce (or avoid) software costs and so are potentially good business decisions.

This month, my contrarian idea was to revisit DB2 data compression. Since the early 90s, DB2 data compression on z/OS has taken advantage of the mainframe hardware support for data compression. The result is that there's relatively little CPU overhead associated with doing data compression, and there are potentially significant performance gains as there are fewer I/Os required to read a given number of rows. Since the data is kept compressed in the buffer pools, the amount of memory needed to achieve a given buffer hit ratio is also reduced. Doing fewer I/Os and achieving better buffer hit ratios are definitely very positive performance changes.

However, software costs are not based on the amount of memory we have on the machines or the number of I/Os that we do. "Relatively little" CPU overhead is not zero overhead. Since we pay for software based on CPU and not memory or I/O, my thought was perhaps we could trade disk and memory for CPU. That is, perhaps we could add memory to super-size the buffer pools, add disk (if necessary) and then turn off data compression to save some CPU. In the ideal world, if we could avoid a planned upcoming CPU upgrade by spending money on disk and memory, this could very well be a good trade-off. Part of the theory was that because the data is compressed in the buffer pools, that small decompression overhead is paid every time the data has to be read from the buffer pool. Or at least that is my supposition: I couldn't find any references to exactly when DB2 decompresses the data, just references to it being compressed in the buffer pools.

We compress almost all of our DB2 data. For our largest application, the total savings across the application is about 44%, with the savings in some tables over 90%. For testing purposes, I selected a fairly busy and important table that had a compression savings of about 80%. While that's more than average, it was closer to the common savings for some of the busier tables, so I felt it was a fair representation. I then had my friendly DBA load an instance of that table into

the DB2 subsystem that we use for performance testing. I was the only one using the subsytem at the time so I could measure the CPU time consumed in the subsystem address spaces as well as the CPU time attributed directly to my test jobs. Half of the partitions had compression enabled and half of the partitions did not. This allowed me to run the same queries against compressed and uncompressed partitions just by changing the predicate slightly.

What I found was not terribly surprising: there was a measurable reduction in CPU time attributed to the test jobs when they were running against uncompressed partitions. This was most significant in jobs that had to scan a significant number of data pages. However, this CPU time savings in the job was offset by an increase in CPU time in the DBM1 address space to do more I/O to prefetch those uncompressed pages. In jobs where the same small subset of pages was read repeatedly, achieving a buffer hit ratio nearing 100%, the uncompressed CPU time advantage was much slimmer--within the normal range of variation. Given those results, I concluded that using DB2 data compression still makes as much sense as it did the last time we tried to quantify the cost or savings, which was likely at least 15 years ago.

As part of doing this, I tried to calculate the decompression overhead per getpage. My best guesstimate for that is on the order of 5-10 CPU microseconds, on our z10 EC 5xx machines. Measurements that fine, that can only be indirectly measured by comparing similar but slightly different executions, have a relatively high degree of uncertainty around them. So you should not put much importance on that number, but I thought it was interesting enough to share. Also note that I didn't try to measure the impact of compression on update and insert activity: that overhead is said to be much higher, but since the vast majority of the activity at our peak times is read activity, I was primarily concerned with that.

On the one hand this was a negative result in that my idea for saving CPU by adding memory and disk turned out to be mostly worthless. However, as often happens, the process of intently investigating the data had value itself as I discovered that we had some production tables that were not being reorged as frequently as they should be.[1] So we are now revisiting the process that selects objects for reorgs. Fixing that process will result in some modest performance and CPU improvements for at least some queries.

So I have a couple of lessons this month: First, DB2 data compression is good: the CPU overhead really is pretty minor. Second, chasing rabbit trails may not lead you to the rabbit, but you may find something else useful along the way that you wouldn't have otherwise come across.

---

[1] For some potentially useful thresholds for determining when to reorg, see chapter 39 in the DB v9 "Performance Monitoring and Tuning Guide" or chapter 40 in the DB2 v10 "Managing Performance" manual.

As always, if you think I've got it all wrong, or have ideas for improvements, please email me at ~~sachapman@aep.com~~ scott.chapman@epstrategies.com and let me know!