What I Learned This Month: IBM DB2 Analytics Accelerator

Scott Chapman
American Electric Power

We're currently working on getting our IBM DB2 Analytics Accelerator (IDAA[1]) up and running. We're still figuring many things out, so "What I'm *Learning* This Month" would be a more accurate title, but I thought I'd write up what I've learned so far.

The IDAA is a hardware appliance (based on the Netezza technology that IBM acquired in 2010) that is managed and exploited by DB2 on z/OS to greatly accelerate certain queries. The primary targets are analytical queries that might traditionally be handled with a data warehouse solution because they would perform poorly in the traditional operational database. While data warehouse solutions have become ubiquitous, there are some issues with them. In particular, they represent a second copy of corporate data to secure and manage. And you need some sort of mechanism to keep that second copy up to date with the changes in the operational data store. Typically there is also a fair bit of work to design the data warehouse to summarize and transform data such that particularly interesting business queries can be answered quickly. Of course over time, the business requirements may change, necessitating changes to the data warehouse.

In contrast, the idea behind the IDAA is to allow DB2 to manage all of that complexity and allow all queries to run against the operational database. DB2 will decide whether to satisfy the queries using traditional z/OS resources or using the IDAA resources. This could be a tremendous simplification of the environment relative to the traditional data warehousing solution.

Additionally, because of the highly parallel nature of the IDAA, typical IDAA eligible queries may execute an order of magnitude (or more) faster than they would if they used z/OS resources. Queries that are running on the IDAA are also not consuming z/OS resources, which may be a potential cost savings and/or may allow other z/OS workloads to perform better.

All and all, it sounds great. So far it does perform well in our initial limited tests. I've done scans of 100-million row tables in just a couple of seconds. But there are some caveats and some things we're still working out.

One of the first issues we ran into was the hardware requirements: the connections between the mainframe CECs and the IDAA appliance must be 10GbE network connections. There were numerous discussions about whether these could be direct connections or whether we were required to install switches and whether or not those switches needed to be dedicated to this purpose. Both the 10GbE cards and the switches are fairly expensive. This caused us a lot of consternation. It would have been a lot easier if the IDAA was simply orderable

[1] I will refer to it herein as the IDAA as a handy abbreviation, even though IBM is no longer branding as that due to a request from International Doctors in Alcoholics Anonymous.

with an embedded switch, but that's not an orderable feature. In the end we did use non-dedicated switches, although they did need a configuration because they initially didn't have jumbo frame support enabled.

On the software side, DB2 v10 New Function Mode is required, at least if you're going to be using the incremental update feature to keep the IDAA tables in sync with the DB2 tables. We're still rolling out NFM across our environments, but we've been able to start experimenting with the IDAA in one of our first NFM environments. In some simple tests we've seen very simple queries that would take over a minute to do a tablespace scan in DB2 on z/OS run in less than 10 seconds in the IDAA. Those are encouraging results, even if they are for simplistic cases and relatively small tables.

One unexpected issue that we ran into is that the IDAA oddly can't handle all the valid data types that DB2 can. In particular, DB2 allows 24:00:00 as a valid time, but that is not valid in the IDAA. IBM has some patches that they can apply to the IDAA to make it more tolerant of 24:00:00 in the DB2 tables, but those fixes involve changing the value to 23:59:59.999999. Whether that's acceptable or not will be application-dependent. If the application is using 24:00:00 to indicate "end of day", changing the value may not be acceptable. For now we changed the data in the DB2 tables and the application changed their code to avoid storing 24:00:00. We're still debating over applying the IDAA patches, but I think we probably will because without them in place, the incremental update data replication will break if a value of 24:00:00 shows up in the future.

Breaking replication could be a significant problem because the use of the IDAA is truly transparent: once a table has been initially copied to the IDAA and enabled for acceleration, any dynamic SQL query for that table may be executed in the IDAA. But since the data in the IDAA is a copy of the data in DB2 on z/OS, the IDAA copy is by necessity somewhat out of date. If replication is enabled, the IDAA data may only be seconds behind DB2. Even a data latency[2] of a few minutes is probably adequate for the vast number of business queries. However, latencies of hours could be problematic for a large number of business queries.

My concern is that we've seen instances where we've been able to break replication for a table (more on that in a moment), but the table remains enabled for acceleration. If the replication failure isn't noticed and handled by somebody in a timely fashion, it's entirely conceivable that we could have queries running against data in the IDAA that's hours old. In some cases that could be very bad. Fast answers are nice, but correct answers are an absolute requirement!

IBM provides some sample code[3] to monitor latency in the Redbook "Hybrid Analytics Solution using IBM DB2 Analytics Accelerator for z/OS 3.1". It seems likely we'll build some sort of automated checking to make sure that the replication process is always running correctly and disable IDAA acceleration if

---

[2] The time between updates being committed in DB2 z/OS and being applied to the IDAA's copy of the data.

[3] See Appendix E in http://www.redbooks.ibm.com/redbooks/pdfs/sg248151.pdf

it's not. You can enable or disable acceleration for individual tables: it would be nice if there was another setting that was "enable if the data is no more than x seconds old".

As for why we were able to break replication, that has to do with the amount of storage that we gave the CDC (Change Data Capture) task. The CDC task uses a staging area in memory to hold uncommitted changes. If you have SQL statements that update or delete millions of rows of data in a single unit of work, you may find that you need to make that area multiple GBs. The amount of storage required is a function of the row length and the number of rows affected. Updates require twice as much memroy as both the before and after image is staged.

Determining the total number of potentially uncommitted inserted, updated, or deleted rows that might be in flight at any given time is a challenging task. However my friendly DBA pointed out that for our largest application we don't allow lock escalation to occur automatically, so the total number of rows or pages that have been updated but not committed is limited to the maximum number of locks that a thread can have.

With that idea in mind, I did some rough worst-case calculations which suggested that a staging area of about 1.2GB would be sufficient for our normal workload. There are a lot of assumptions built into those calculations and we do have certain occasional processes that lock a whole table to do a massive update, which can drive the need for a much larger staging area. In fact during testing we found we needed a 1.7GB staging area when we were running a process that deleted or inserted 3 million rows with a single commit. We decided to make the staging area 3GB and monitor and measure it to make sure that we're not running close to that limit. I expect that we would have been fine with the 2GB default, at least until something came along and impacted several million rows at once. I'm glad that we found this during testing instead of finding it by surprise in production some weeks or months after implementation.

So that's what I've learned so far about the IDAA. I'm looking forward to working through the questions that I still have and doing some tests with larger volumes of data and more sophisticated queries that better represent some of the problems we expect that it will solve for us. Performance will also likely get more "interesting" once we start sending multiple queries to it simultaneously.

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