What I Learned This Month: DDF. Enclaves. Again.

Scott Chapman

American Electric Power

You may be thinking, "won't he ever stop writing about DDF and enclaves?" Well, as soon as I get it all figured out I will, but I keep learning new things. This is in part because things have changed a few times over the past couple of years.

You may remember that a couple of months ago I wrote that I thought there were more work-dependent DDF enclaves than I expected on my system.[1] I was confused because the vast majority of the documentation indicates that work-dependent enclaves are used for DDF transactions that use query parallelism.[2] Query parallelism is a process by which DB2 breaks certain queries into multiple pieces and runs multiple concurrent tasks either on multiple CPUs on the same system (CPU-parallelism) or across multiple systems (sysplex-parallelism).

Parallelism isn't necessarily enabled by default, and in fact we have it disabled for many workloads. While it definitely can help individual transactions, it does so by making those transactions more intensive for a shorter period of time. And there is some additional overhead to split the query apart and then merge the results back together. These trade-offs don't always result in a net benefit to the system overall. Whether it makes sense depends on the workloads, their relative importance, and how much spare capacity you have.

So when I saw more work-dependent enclaves showing up, I was somewhat surprised because I thought we didn't allow parallelism for any DDF transactions. Indeed when I checked the DB2 monitors for a few of the transactions, it indicated that there was no parallelism involved. That was confusing: either the monitor was not correctly indicating parallelism, or work-dependent enclaves were being used for some other purpose. (Perhaps stored procedures, but there was no indication of stored procedure usage either.)

Recently, I was running some of my own code that used a JDBC connection into DB2 to insert data from a file on my PC. While I'm always interested in performance, I'm especially interested when it involves my own code. Since I know the design intention and the implementation details, it's an excellent learning opportunity to see the impact of particular design and coding choices.

I was surprised to see my transactions generating work-dependent enclaves. In fact, there were two enclaves for each transaction: one independent enclave and the work-dependent one that was apparently spawned from the independent one.

---

[1] See the end of: http://www.cmg.org/measureit/issues/mit93/m_93_5.pdf

[2] See for example the original APARs that added support for work-dependent enclaves for DDF work: PK87913 and OA26104.

Although I did check the DB2 monitor, I really didn't need to. I knew there was no opportunity for the inserts to use parallelism.

It seems that DB2 is now creating two enclaves for every DDF transaction. The independent enclaves seem to run 100% zIIP-eligible and the work-dependent enclave runs 0% zIIP eligible and about 60% of the time ends up on the independent enclave. You can see this in this sample snippet from the SDSF enclave panel:

NAME	Original	Type	SSType	Status	SrvClass	CPU-Time	zIIP-NTime	zICP-Time
1D400136191	YES	IND	DDF	ACTIVE	DDFDEV	9.04	8.91	0.09
1EC00136192	NO	WDEP	DDF	ACTIVE	DDFDEV	5.99	0.00	0.00

Although I haven't found documentation explaining why this was done, it does solve the issues that I had previously raised with running transactions either 100% or 0% offloaded[3]. I'm not sure when in the last year this functionality showed up on our system, but I have been wondering about the relatively high number of DDF enclaves on our systems for several months.

While this solves the bigger issue, it does raise a couple of new ones. First, there must be at least a small amount of overhead associated with splitting work across two enclaves. Second, it does make things messier when looking at the RMF III or SDSF enclave panels: there are now twice as many enclaves reported for the same amount of work. If you're looking for the most intensive transactions, you have to mentally sum up the two enclaves that make up each transaction. That's inconvenient and error-prone.

I'll reiterate what I said months ago: all of this effort to offload a specific portion of DDF processing to the zIIPs seems like unnecessary trouble. IBM should do for DDF work what it did for Java work: make it all eligible. This would simplify IBM's code, make the rules simpler to understand and improve performance. It would only hurt IBM revenue to the degree that customers are using variable workload pricing and their peak usage is driven by significant amounts of DDF work. On the other hand, it would eliminate many of the arguments against running queries against mainframe databases and much of the reason for offloading data from DB2 to other platforms for later querying.

IBM's IDAA offering for a select type of workload[4] could also be said to be a precedent for making a certain class of SQL 100% offloaded. Making the DDF zIIP offload 100% and only using work-dependent enclaves where they make logical technical sense (e.g. for parallel queries) would be a customer-friendly decision and I think it would be a net benefit to IBM in the long term.

So if you have noticed more DDF enclaves on your systems recently now you know why: every transaction now generates two enclaves. And if you think this is getting too complicated, consider explaining to IBM how simpler alternatives

---

[3] See: http://www.cmg.org/measureit/issues/mit82/m_82_2.pdf

[4] Primarily analytics. See http://www.ibm.com/software/data/db2/zos/analytics-accelerator/

would benefit IBM and their customer! As always, if you have questions or comments, you can reach me via email at sachapman@aep.com.