DSNZPARMs: DB2 System Configuration Parameters
By Craig S. Mullins

There are multiple types of IT professionals who deal with, use, and manage DB2 for z/OS on a daily basis. The vast majority are end users, followed by application developers and programmers. Then there are the DBAs, of which there are many varieties: application-focused, system-focused, performance-focused, and even jacks-of-all-trades. And finally we have the system programmers, those who work with installation and system tuning.

Of all these constituents, usually only the system programmers and *some* of the DBAs get involved with system-level performance and configuration issues. But this topic is getting more complex and difficult to manage all the time, so let's take a look at some of the issues.

Probably the most important aspect of system configuration and tuning involves understanding and managing DSNZPARMs, aka ZPARMs. These are the system parameters that control the overall behavior of a DB2 subsystem.

There are seven macros that, once assembled, produce the DSNZPARMs: DSN6ENV, DSN6ARVP, DSN6LOGP, DSN6FAC, DSN6GRP, DSN6SYSP, and DSN6SPRM. I don't want to get into an explanation of how to set ZPARMs or what they are all in this column, though.

So far, so good -- but keep in mind that there are ***literally*** hundreds of these parameters, with more being added all the time. And the documentation on them is limited.

Some Example DSNZPARMs

The sheer volume of DSNZPARM parameters can be overwhelming. Oh sure, some of them are more visible than others. For example, many people know about the locking parameters (NUMLKUS – maximum locks per user; NUMLKTS – maximum locks per user per table space) and some of the others like:

- CTHREAD – maximum users
- CONDBAT, MAXDBAT – maximum remote connections, active
- IDFORE, IDBACK – maximum TSO, batch
- DSMAX – maximum number of open data sets

These parameters have been in DB2 for ages and many of us have had exposure to them. But every new DB2 release adds more parameters and options, as well as changes old parameters, so we have to tango with many, many more ZPARMs than before. As such, sometimes we just go with the defaults and hope for the best.

But this is not really a viable option in this day and age when we need to squeeze every last drop of performance out of our systems. Setting even just one ZPARM

inappropriately for your system can have a huge negative impact on performance; and if you just take the defaults, you may have more than one thing set sub-optimally.

EDM storage is set using DSNZPARMs. In the early days of DB2, there was a single EDM pool. But over the course of the past few releases, IBM has broken the EDM into a series of separate pools. The EDM pools are used to maintain DBDs, cursor tables (for plans), package tables, authorization cache, the dynamic statement cache needed by executing SQL statements.

As of DB2 10 for z/OS, all the EDM pools are allocated above the 2GB bar. These EDM pools are

- Pool for packages and plan (PTs and CTs)
- Skeleton pool for packages and plans (SKCTs and SKPTs)
- Prepared-statement cache pool for dynamic SQL
- Database descriptor (DBD) pool

The size of the various EDM pools is specified in the DSNZPARMs and must be determined before starting DB2. For new DB2 subsystems, letting the DB2 installation process calculate the size of the EDM pools is best. For existing DB2 subsystems, you can modify the EDM pools based on information about your environment.

It is important to get the sizing of the EDM structures correct for your mixture of programs, dynamic SQL usage, and database objects. If the EDM structures are too small you run the risk of slowing down your applications waiting for DB2 to load skeletons to runs your programs or DBDs to access your data.

The EDMPOOL DSNZPARM controls the size of the EDM structure that contains CTs and PTs. It can range from 0 to 2097152. As of DB2 V10, EDM does not allocate storage below the bar except to process a package or plan that was last bound or rebound prior to Version 10. If you migrate to Version 10, override the default setting with the EDMPOOL value that you currently use in the previous release.

The pool for skeletons is controlled using the EDM_SKELETON_POOL DSNZPARM and can range from 5120 to 2097152. The default is 10240.

To estimate the size of the EDM pools for CTs and PTs and skeletons, you need to know the maximum number of concurrently executing plans and packages, as well as the average plan and package size. You can obtain the average plan and package size by querying the DB2 Catalog as follows:

```
SELECT   AVG(PLSIZE)
FROM     SYSIBM.SYSPLAN

SELECT   AVG(PKSIZE)
FROM     SYSIBM.SYSPACKAGE
```

Next we turn our attention to the EDMDBDC DSNZPARM, which controls the size of the EDM DBD cache. It can range from 5000K to 2097152KB. The default is 23400K. To estimate the size of the EDMDBDC, you need an estimate of the number of concurrently accessed DBDs and the average DBD size.

To arrive at the average DBD size, you must know the average number of columns per table and the average number of tables per database. For an existing DB2 implementation this information can be found in the DB2 Catalog.

To determine the average number of concurrent DBDs, you must understand each application's database use. If an application that typically uses three databases is much more active than another that uses 12 databases, you must factor this information into your sizing strategy. Obtaining this information can be difficult, so you might need to estimate.

As your DB2 usage patterns change, plan and package sizes can grow, necessitating EDM pool growth.

Another interesting aspect of DSNZPARMs is to control checkpoint frequency in DB2 subsystems. DB2 periodically must take a system checkpoint. The checkpoint contains information on the currently open unit of recoveries within DB2, all open page sets, a list of page sets with exception states, and a list of page sets updated by any currently open unit of recovery. Also, buffer pool pages that need to be externalized are written out at system checkpoint.

The frequency of checkpointing is controlled using the CHKFREQ parameter. So it is important to set this ZPARM appropriately, as well as your buffer pool deferred write thresholds (DWQT). Taking appropriate checkpoints speeds up the DB2 restart process. But taking a checkpoint also causes some minimal contention within DB2 and will also increase CPU usage by DB2.

The default for this parameter used to be 500,000, which meant that DB2 would take a checkpoint after writing 500,000 log records. But this was not ideal for many environments. Do you know how much time it takes to write that many log records in your shop? Probably not.

So over the course of the past several releases, IBM changed things. As of DB2 V7, the CHKFREQ parameter was modified to be able to accept a number of minutes, instead of the number of log records written, if you so choose. This helped because the general advice was to take a checkpoint every 5 minutes during peak processing. So now we can set it to 5 minutes instead of try to figure out the number of log records that would be processed.

Today, as of DB2 11 for z/OS, we can use the number of log records processed (CHKLOGR) and number of minutes since the last checkpoint (CHKFREQ) or even both if so desired.

Let's take a look at another interesting system parameter: STATIME. This ZPARM is used to set the number of minutes between statistics trace collections. There are a lot of interesting statistics collected by the STAT trace and it makes sense to set this value to a small interval. In earlier versions of DB2, the default for STATIME was 1,440 – and then it was dropped to 60. As of DB2 V10 the default is 1, and that is a good value for most installations.

Additionally, keep in mind that as of DB2 Version 10, the STATIME subsystem parameter applies only to IFCIDs 0105, 0106, 0199, and 0365. IFCIDs 0001, 0002, 0202, 0217, 0225, and 0230 are no longer controlled by STATIME, and the corresponding trace records are written at fixed, one-minute intervals.

Before we go down a rabbit hole trying to describe more of these parameters, remember that there are hundreds of DSNZPARMs, all with different impacts, effects, and concerns. With that in mind, instead of looking at more individual parameters, let's delve into some of the issues involved in setting and changing ZPARMs.

Changing DSNZPARMs

In the olden days, prior to DB2 Version 7, the only supported way to make changes to DSNZPARMs was to change the macro keywords to new values, assemble the modified macros, and link-edit them into a new load module. DB2 had to be stopped and restarted to load the newly created load module into storage so changes could be activated. Obviously, this is a less than desirable method for making changes, especially if it is the only one at your disposal. Indeed, some of the third party tool vendors built products to enable many DSNZPARM changes to be made without recycling DB2.

As of DB2 V7, though, we have the -SET SYSPARM command at our disposal. This command enables a system programmer or DBA to replace the load module that DB2 is using for DSNZPARM value with another load module. So it becomes possible to using –SET SYSPARM to get DB2 to use new DSNZPARM values without bring DB2 down. Of course, not every DSNZPARM value is supported using this method, but it is a step in the right direction.

So the bottom line here is that some DSNZPARMs can be changed online, while DB2 is up and running, and others cannot. If you have examined an issue that can be resolved by modifying a DSNZPARM, be sure to investigate whether or not the parameter can be changed online.

A Lot More to Learn

Of course, this article just looks at the tip of the iceberg regarding DB2 system parameter configuration and DSNZPARMs. Other factors that should be considered include:

- Who is permitted to perform DSNZPARM changes?
- When and how should we go about modifying DSNZPARM values?

- Do you know about the opaque and hidden DSNZPARM values that are available which can be used to tune your DB2 environment (but are not documented)?
- What change management procedures are required for DB2 system parameter changes? Is specific documentation required for data governance and regulatory compliance mandates?
- How do you document what parameters gets changed, along with when, and why?

Indeed, there is a lot to consider in terms of what is happening with DB2 for z/OS under the covers… So end users and application folks, aren't you glad there are people that worry about this in your shop so you don't have to? There are people worrying, right?