CONFIGURING DB2 FOR Z/OS BUFFER POOLS

Guest post by Lynn Gros, Lead Quality Assurance Representative

Second in a series of posts.

Last week, we reviewed some buffer pool basics. This week, we'll discuss how to configure buffer pools for optimal performance.

Most Db2 users revisit buffer pool sizing and thresholds over time to make sure they meet performance goals. The IBM Db2 on z/OS Installation Guide provides information on initial sizing for Db2 buffer pools, and it is often the best place to start for new applications or subsystems. Once applications are in test or production, you can collect metrics that may lead you to consider changes to the buffer pool configuration.

General considerations

Use the simplest approach that effectively meets performance requirements. Your major task is determining Db2 object placement in the various pools. This effort will lead to the physical characteristics of the buffer pool configuration for your Db2 subsystem.

A good starting point is to consider the various Db2 object types you deal with on a day-to-day basis, including:

- System – Db2 catalog and directory objects (DSNDB01 and DSNDB06)
- Work – Storage used for temporary files and tables required for a multiple Db2 functions, including global temporary tables, Db2 sort work files, and other functions. Because these pagesets tend have high I/O volumes, consider page fixing for these pools. Db2 9 merges the

WORKFILE and TEMP databases from Db2 V8 into a single WORKFILE database. Work file enhancements in Db2 10 reduce the CPU time for workloads that execute queries that require the use of small work files, and improve scalability. Db2 10 supports partition-by-growth table spaces in the WORKFILE database, and it provides more opportunity to use in-memory work files than Db2 9.
- Large objects – LOB/XML structures.

- LOBs should not share a buffer pool with other kinds of data. Set the DWQT (deferred write threshold) to 0 for these objects. This allows updates to happen continuously (trickle writes) in the background to avoid massive writes at Db2 checkpoints.
- XML table spaces use uses 16K buffer pools. Place these objects in dedicated pools. Use a buffer pool other than BP16K0.
- Application table spaces and index spaces

Place WORKFILE and TEMP database objects in dedicated buffer pools. Typically, workfile pagesets are accessed sequentially and are short-lived objects. Db2 9 will try to make greater use of 32K workfile table spaces, so increase the size of the 32K buffer pool used for the WORKFILE database and increase the number of 32K page workfile data sets.

It is important to understand Db2 application object characteristics when deciding on buffer pool placement. Grouping objects with similar characteristics into the same buffer pool can be a good approach to meet your performance objectives. Consider these characteristics:

- Business priority – Perhaps the most important information you need is how critical the application is to your business. Consider placing critical application objects that require maximum performance into dedicated buffer pools to avoid resource contention with less-important application objects.
- Random or sequential access – Is the predominant level of GETPAGE activity random or sequential? Place objects that are accessed sequentially most often in a pool with a higher VPSEQT value; place objects with mostly random access in a pool with a lower VPSEQT value. If the accesses are split evenly between random and sequential, use the default values as a starting point;
- Activity against this object – How busy is this object? Activity is measured in GETPAGEs per second against the object. A very busy object might cause buffer pool pages to be stolen from another object that is not as busy but more important to the business.
- Update frequency – What is the update rate for the object? Is this a heavily updated Db2 page set? In general, table spaces and index spaces that are frequently updated should be separated from infrequently updated objects to allow optimum buffer pool sizing and thresholds.
- Object size – How big is this Db2 object in pages? If the object is very large and the access is very random, the chances of reusing a page that's already in the pool will be minimal, so having a very large pool for the very large object may provide no benefit to application performance.

It can be difficult to collect many of these characteristics, especially for a new application. Use an educated guess as a starting point. In test environments, use an isolated set of buffer pools as you begin testing a new application or one that is undergoing major changes. This will give you more complete control of the testing environment where you can monitor buffer pool performance more easily.

Db2 application objects are a much more complex consideration, but here are some general guidelines:

- Don't place application objects in BP0, BP8K0, BP16K0 or BP32K. Db2 catalog and directory objects are placed these pools. Set default buffer pools by modifying these values on the DSNTIP1 installation panel. It's a good idea to specify an explicit buffer pool assignment for your objects; however, if you are allowing implicit object creation, you can't directly specify what pools to use. Changing the installation defaults for user data and indexes will help enforce your strategy for object placement.
- Assign indexes to different buffer pools than table spaces for concurrency and long residency times. Indexes are typically accessed differently from the tables they index - and with greater frequency. The number of buffers required for average working set size of an index space is usually smaller than its corresponding table space/tables, and you want to ensure buffer pages used by index pages remain resident and not get flushed during heavy table space scan activity. Like table spaces, separate frequently updated indexes from infrequently updated indexes to allow optimum buffer pool sizing and thresholds.

Next week, we'll look at metrics.

> *The postings in this blog are my own and don't necessarily represent BMC's opinion or position.*