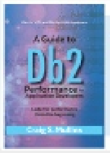

The Resource for Users of IBM zSeries & S/390 Systems
z/JOURNAL

June / July 2011

zData Perspectives

Partitioning Pros and Cons

by Craig S. Mullins

Before deciding whether or not to use a DB2 partitioned table space, it is wise to first weigh the pros and cons. I will try to make the following list of advantages and disadvantages applicable to partitioning in general; that way it applies to all types of partitioning (index-controlled, table-controlled, and universal partition-by-range).

Advantages of a partitioned table space:
Each partition can be placed on a different disk volume to increase access efficiency. Of course, this is more difficult (if not impossible) to accomplish with RAID storage devices, so this advantage is not generally applicable for shop's using modern

DB2 creates a separate compression dictionary for each table space partition. Multiple dictionaries tend to cause better overall compression ratios. In addition, it is more likely that the partition-level compression dictionaries can be rebuilt more frequently than non-partitioned dictionaries. Frequent rebuilding of the compression dictionary can lead to a better overall compression ratio.

The REORG, COPY, and RECOVER utilities can execute on table spaces at the partition level. If these utilities are set to execute on partitions instead of on the entire table space, valuable time can be saved by processing only the partitions that need to be reorganized, copied, or recovered. Partition independence and resource serialization further increase the availability of partitions during utility processing.

disk storage arrays.

Partitioned table spaces can be used to store large amounts of data. The maximum size of segmented table spaces is 64GB.

START and STOP commands can be issued at the partition level. By stopping only specific partitions, the remaining partitions are available to be accessed thereby promoting higher availability.

Free space (PCTFREE and FREEPAGE) can be specified at the partition level enabling the DBA to isolate data "hot spots" to a specific partition and tune accordingly.

Query I/O, CPU, and Sysplex parallelism enable multiple engines to access different partitions in parallel, usually resulting in reduced elapsed time. DB2 can access non-partitioned table spaces in parallel, too, but partitioning can optimize parallelism by removing disk contention.

Table space scans on partitioned table spaces can skip partitions that are excluded based on the query predicates. Skipping entire partitions can improve overall query performance for table space scans.

By mixing clustering and partitioning you can design to decrease data contention. For example, if the table space will be partitioned by DEPTNO, each department (or range of compatible departments) could be placed in separate partitions. Each range of departments is in a discrete physical data set, thereby reducing inter-departmental contention due to multiple departments coexisting on the same data page.

You can further reduce contention by creating data partitioned secondary indexes (DPSIs). Prior to V8, some contention remained for data in non-partitioned indexes. Defining a non-partitioned secondary index (NPSI) on a table in a partitioned table space causes you to lose some of the benefits of partition-

Modifying partitioning details, changing partition key ranges and rotating partitions is more flexible than ever with online schema change support.

Of course, there are disadvantages of partitioning, too. For example:

Only one table can be defined in a partitioned table space. This is not really a disadvantage, merely a limitation.

Prior to DB2 V8, updating the partitioning key was problematic. Before the limit key values could be updated the PARTKEYU DSNZPARM parameter had to be set to YES. And if updates were allowed, in all likelihood they ran slowly. If PARTKEYU was set to NO, you had to DELETE the row and then re-INSERT it to change a value in a column of a partitioning key. But things have changed. As of V8 a partitioning index is no longer required. Furthermore, any partitioning key can be updated without worrying about the PARTKEYU parameter, which is no longer even supported.

The range of key values for which data will be inserted into the table should be known and stable before you create define the limit keys for the partitioning index. These ranges should distribute the data throughout the partitions according to the access needs of the applications using the data. If you provide a stop-gap partition to catch all the values lower (or higher) than the defined range, monitor that partition to ensure that it does not grow dramatically or cause performance problems if it is smaller or larger than most other partitions.

Summary

The bottom line is that more and more of your DB2 table spaces will be partitioned. The advantages or partitioning are increasing and the drawbacks are diminishing. This trend is furthered with the advent of universal table spaces and the deprecation of support for simple table spaces. Learn the level independence for utility operations because access to an NPSI is not broken apart by the partitioning scheme.

benefits of partitioning today, and use it to your advantage in your DB2 systems.

From zJournal, June / July 2011.

DB2PORTAL.com