

ANALYZING DB2 DATA SHARING PERFORMANCE PROBLEMS

Donald R. Deese
Computer Management Sciences, Inc.
6076-D Franconia Road
Alexandria, VA 22310

DB2 data sharing offers many advantages in the right environment, and performs very well when DB2 is properly tuned. DB2 data sharing can significantly increase capacity, it offers increased availability, and it provides a flexible operating environment. In certain environments, DB2 data sharing can dramatically improve performance of DB2 subsystems (e.g., when most data are shared and most of the shared data can fit into group buffer pools, data sharing members do not have to continually reference DASD). On the other hand, remarkably unsatisfying DB2 performance can be the result if DB2 data sharing is poorly tuned. This paper describes common problems with DB2 data sharing, and offers alternative solutions to the problems.

1. INTRODUCTION

This paper is in two parts. The first part gives a brief overview of DB2 data sharing concepts and operation. The second part discusses techniques that can be used to analyze common performance problems with DB2 data sharing, and presents alternatives that can be used to solve the performance problems.

This document does not describe all aspects of DB2 data sharing, nor does it describe all performance problems that could occur in a DB2 data sharing environment. Such a detailed discussion would require a formal seminar on DB2 data sharing, and is beyond the scope of this document.

Rather, this document describes aspects of DB2 data sharing that are necessary to understand the implications of common performance problems, it describes the most common performance problems, and it describes alternate solutions to the common problems. DB2 data sharing environments certainly exist in which these descriptions would be inadequate, but this document should prove useful for most DB2 data sharing environments.

In particular, the document does not describe much of the analysis related to coupling facility performance problems. The referenced presentations by Joan Kelley

and Linda August provide excellent descriptions of coupling facility performance analysis.

Also, for simplicity in the discussion, the performance implications of duplexed coupling facility structures are not discussed. Discussion of these performance implications will await a future paper.

2: DB2 DATA SHARING OVERVIEW

DB2 data sharing provides the ability of two or more DB2 subsystems to directly access and change a single set of data. Up to 32 DB2 subsystems can share the data, and the subsystems function as members of a *data sharing group*.

The sharing of data requires that a mechanism be in place to ensure that data integrity and consistency are maintained. If one member of the DB2 data sharing group should change any shared data, the mechanism must make sure that all other members of the data sharing group are advised that their copy of the data is no longer valid.

2.1: Group Buffer Pools

The primary mechanism that is used to ensure data integrity, aside from the basic DB2 locking algorithms provided by Internal Resource Lock Manager (IRLM), is the use of group buffer pools that reside in a coupling facility. Group buffer pools are shared by all members of the data sharing group, and there is a one-to-one mapping between group buffer pools and virtual buffer pools (e.g., Group Buffer Pool 6 corresponds to Virtual Buffer Pool 6).

Permission is granted to CMG to publish this document in its publications. Computer Management Sciences, Inc. retains its right to distribute copies of this document to whomever it chooses.

CPEExpert is a trademark of Computer Management Sciences, Inc., Alexandria, VA 22310. All other trademarks are the property of their respective owners.

©Copyright 2001, Computer Management Sciences, Inc., Alexandria, VA

The primary purpose of a group buffer pool is to ensure data integrity and consistency among members of the data sharing group.

- If there is *inter-DB2 Read/Write (R/W) interest* for a page set or page set partition, each member in the data sharing group must register its interest in a page from the shared page set or page set partition before the page can be read into the member's local buffer pool. This interest must be registered in a directory entry in the group buffer pool. Consequently, every page in the local buffer pool is registered in the page directory of the corresponding group buffer pool when there is inter-DB2 R/W interest.
- When a page is changed by any data sharing member, the directory entry for the page is updated to indicate that the page has been changed. All members of the data sharing group are then notified that the page has been changed, and that the page in their local buffer pool is no longer valid but must be refreshed if referenced again. In this way, data consistency is maintained across data sharing members.

A secondary purpose of a group buffer pool is to provide improved performance for data sharing members that read pages from shared page sets or partitions.

- A data sharing member can write pages to the group buffer pool when the pages are changed¹, or optionally when the pages have been read from DASD.
- When needed by other data sharing members, pages from shared page sets or partitions can be retrieved from the coupling facility, rather than being read from DASD. Retrieval from the coupling facility normally would be much faster than retrieval from DASD².

As illustrated in Exhibit 1, group buffer pool storage is divided into directory entry storage and data entry storage. This means that, of the total storage allocated to the coupling facility structure containing the group buffer pool, some of the storage is reserved for directory entries and some is reserved for data (page) entries. This division of storage is done when the group buffer pool is created and can be changed via an ALTER GROUPBUFFERPOOL command.

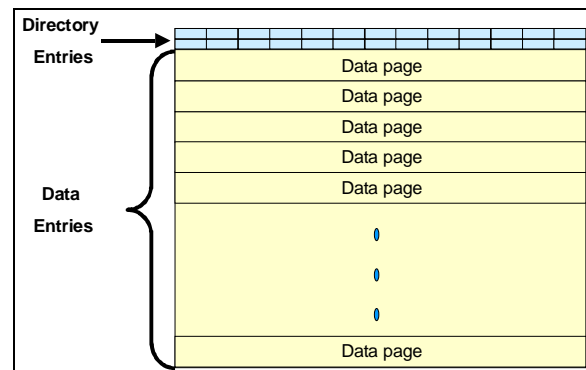


Exhibit 1: Group Buffer Pool Structure

The division is based on a specified **ratio** of directory entries to data entries. The default ratio is 5:1, meaning that there are five directory entries for each data entry in the group buffer pool.

There are more directory entries than data entries because (unless GBPCACHE ALL is specified), only *changed* pages from a virtual buffer pool are placed in the group buffer pool. Unless there is extensive update activity, changed pages account for a relatively small percent of the total number of pages in a virtual buffer pool.

However, a directory entry must be created to account for each page from a shared page set that is in **any** virtual buffer pool or hiperpool of members of the DB2 data sharing group. Consequently, there normally would be far more directory entries than there would be data entries. IBM has decided that, as a default, there should be five times as many directory entries as there are data entries.

Data entries are the same size as the page in the corresponding local buffer pool in a data sharing member (e.g., if the page were defined as 16K in the local buffer pool, the data entry in the group buffer pool would be 16K). Data entries hold the actual data read from DASD or updated by data sharing members.

Directory entries are much smaller (typically about 200 bytes, but the size depends on the CF level). Directory entries hold control information about the data pages they describe.

One implication of this difference in size between directory entries and data entries is that, although there normally are many more directory entries than data entries, the total storage in the group buffer pool that is occupied by directory entries normally is less than that occupied by data entries.

¹Note that prior to DB2 Version 6, **all** changed pages were written to the group buffer pool. With DB2 Version 6, the group buffer pool optionally can be used **only** for data integrity and consistency by specifying GBPCACHE NONE.

²DB2 actually determines the fastest access method, and reads data from the coupling facility or from DASD, depending on which method is quicker.

2.2: Registering Pages

As mentioned earlier, with data sharing and a shared page set, DB2 must register the page with the coupling facility (register with the group buffer pool) if there is *inter-DB2 R/W interest* in the page set. This is done so DB2 can maintain integrity of the data. For example, if one DB2 should change a particular page, the page must be invalidated in the virtual buffer pool or hiperpool of all other members of the DB2 data sharing group.

The page is registered with the group buffer pool before it is read from DASD. If a directory entry exists for the page, the directory entry is updated to reflect that *this* DB2 has an interest in the page. If a directory entry does not exist for the page, a directory entry is created and updated to reflect that *this* DB2 has an interest in the page.

Creating a directory entry means that storage must be available in the group buffer pool to hold the directory entry. Under certain situations, there may be no directory entry storage available in the group buffer pool and the read fails because of lack of storage (this problem will be discussed later).

Exhibit 2 illustrates example actions that can occur when DB2 subsystem *DB2A* (1) registers interest in a page and (2) reads the page.

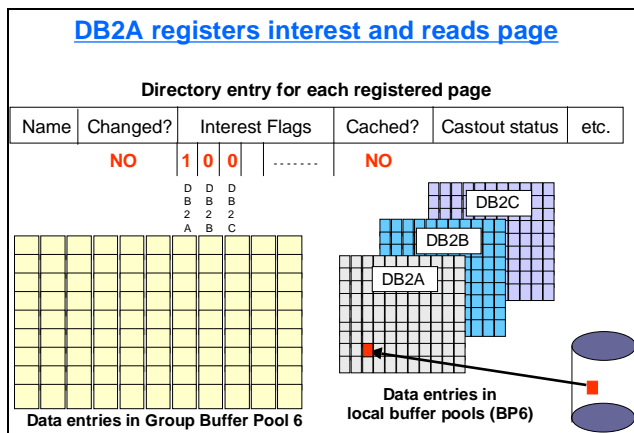


Exhibit 2: DB2A registers interest and reads page

Exhibit 2 shows that the page is not written to the group buffer pool when it is read from DASD. With DB2 Version 6, there are four options with respect to whether (and when) a page is cached in the group buffer pool:

- Only changed pages are written to the group buffer pool with the **GBPCACHE CHANGED** option (GBPCACHE CHANGED is assumed in Exhibit 2 and the following exhibits).

- Pages are written to the group buffer pool as they are read from DASD with the **GBPCACHE ALL** option.
- Only large object (LOB) space map pages are written to the group buffer pool with the **GBPCACHE SYSTEM** option.
- Pages are never written to the group buffer pool with the **GBPCACHE NONE** option (the group buffer pool is used solely to ensure data integrity by implementing page cross-invalidation).

Exhibit 3 shows an example action that can occur when DB2 subsystem *DB2B* wants to read the page that was just read by *DB2A*. When *DB2B* wants to read the page, it registers its interest in the page, and verifies that the page is not in the group buffer pool.

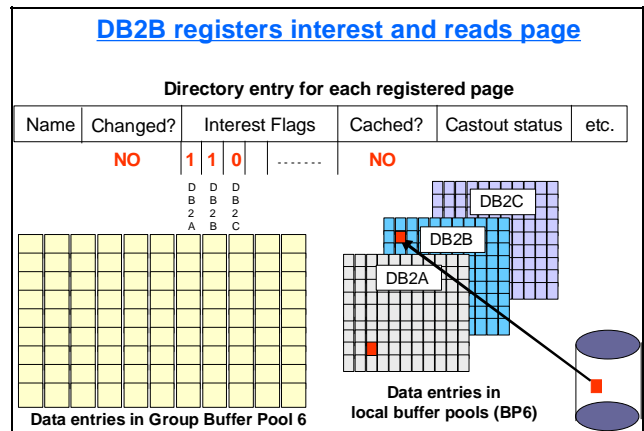


Exhibit 3: DB2B registers interest and reads page

Notice that the page would be in the group buffer pool if GBPCACHE ALL had been specified, even though the page had not been changed.

When a DB2 subsystem needs to reference a page in a DB2 data sharing environment, it looks for the page in the following order:

Step 1: Search the virtual buffer pool. The page might exist in the buffer pool, but might have been marked invalid by DB2 data sharing integrity code. If it is marked invalid, DB2 skips to Step 3.

Step 2: Search the hiperpool buffer pool. The page might exist in a hiperpool both with non-data sharing and with data sharing (unless GBPCACHE ALL has been specified for the page set, in which case a hiperpool is not used). However, the page might be marked invalid in the hiperpool by DB2 data sharing integrity code. If it is marked invalid, DB2 goes to Step 3.

Step 3: Search the group buffer pool. Unless GBPCACHE parameters override, DB2 next checks the group buffer pool for the page. This is done if there is R/W interest in the page set or the partition, or if the page set is defined as GBPCACHE ALL.

Step 4: Read the page from DASD. DB2 reads the page from DASD if the page is not found after applying the above search algorithm.

In our example, the page is not in the group buffer pool. Consequently, DB2B must read the page from DASD.

This is a good illustration of the **potential** benefits of GBPCACHE ALL. Suppose that DB2B regularly reads pages from the table space that it shares with DB2A. If sufficient coupling facility storage were available, and if the application was important, perhaps GBPCACHE ALL should be specified. In this illustration, DB2B would have found the page in the group buffer pool (because it would have been placed in the group buffer pool by DB2A) and would not have to read the page from DASD. A considerable improvement in performance could result, particularly if many pages were read.

2.3: Maintaining Data Integrity

Suppose that DB2A changes the page being discussed. The page must be invalidated in the local buffer pools of all data sharing members that have an interest in the page.

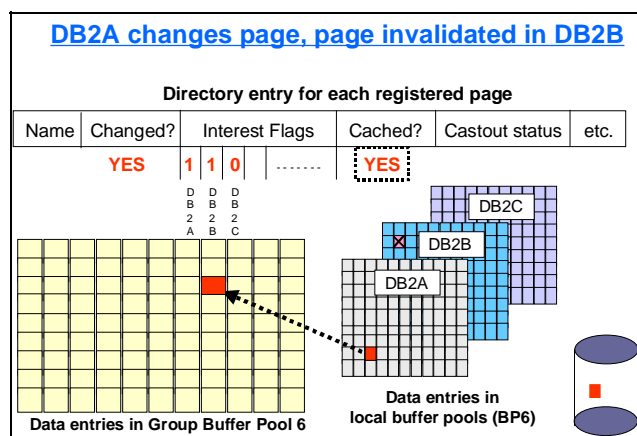


Exhibit 4: Maintaining data integrity

Exhibit 4 illustrates the example actions that occur when DB2 changes a page and invalidates the page in other DB2 subsystems in the data sharing group. In this illustration, the only other data sharing member with an interest in the page is DB2B, and the page is invalidated in DB2B's local buffer.

Exhibit 4 shows that page has been written to the group buffer pool. Actually, this is misleading, since the page probably is still only in the local buffer pool of DB2A (the dashed line and dashed box around YES under "Cached?" indicate that the page might not yet be cached).

In a non-data sharing environment, changed pages in a virtual buffer pool are not normally immediately written to DASD once they are changed. These changed pages are retained in the virtual buffer pool awaiting the possibility that they will be referenced again. The pages remain in the virtual buffer pool until certain thresholds are exceeded or until a DB2 checkpoint is taken. This concept is termed "deferred writes" of the pages.

With data sharing, DB2 still performs deferred writes (i.e., the page is retained in the virtual buffer pool). However, when an update is to a page set for which *inter-DB2 R/W interest* exists, DB2 writes the updated pages from the virtual buffer pool to the group buffer pool, rather than writing the pages to DASD. DB2 writes updated pages from the virtual buffer pool to the group buffer pool when:

- A page is required for update action by another system because there is no conflict on transaction locking (such as with page sets that are using row locking, index pages, space map pages, etc.).
- The transaction commits. When committing a transaction that performed an UPDATE, pages that were updated but not yet written to the group buffer pool are synchronously written to the group buffer pool.
- Normal DB2 virtual buffer algorithms cause the page to be written to DASD (e.g., the virtual buffer pool's DWQT or VDWQT thresholds were exceeded).

2.4: Reading Changed Page

When a DB2 member needs a page of data and finds the page in its local buffer pool (either a virtual buffer pool or a hiperpool), it tests to see if the buffer contents are still valid. If the local buffer contents are not valid, the page must be refreshed, either from the group buffer pool or DASD.

Exhibit 5 illustrates the example actions that occur when a DB2 subsystem wishes to read a page that has been invalidated.

In this example, the page was in the local buffer pool. However, it had been marked invalid by DB2 data sharing integrity code, since it had been changed by DB2A. Consequently, DB2B must refresh its version of the page. In this example, DB2B finds the page in the group buffer pool, and reads the page from the group buffer pool.

In Exhibit 5, DB2A is shown as writing the page to the group buffer pool only when it is required by DB2B. This situation illustrates that even though the page had been changed, DB2 does not necessarily write the page to the group buffer pool immediately on change.

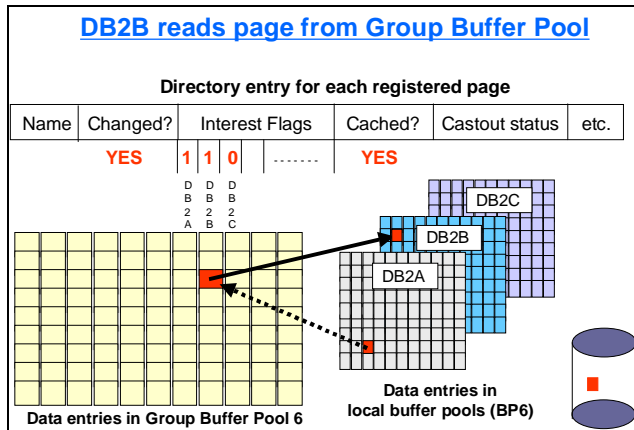


Exhibit 5: Reading an invalidated page

Whether the page is in the group buffer pool when needed by DB2B depends on whether the Deferred Write Threshold (DWQT) or the Vertical Deferred Write Threshold (VDWQT) on DB2A had been reached, whether the transaction on DB2A had committed, how DB2B obtained a lock, etc. Thus, there is a dashed line between DB2A and the page in the group buffer pool, indicating that DB2A might write the page to the group buffer pool only when required to satisfy the request from DB2B.

Under many conditions, the page will reside in the group buffer pool when needed by a data sharing member even though the page had been invalidated. This is because the changed page had been written to the group buffer pool, based on the conditions described in Section 2.3.

Once having been written to the group buffer pool, the changed page will reside in the group buffer pool until castout (castout is described in the next section). When needed by another DB2 data sharing member, the page might not have been castout from the group buffer pool to DASD. Even if the page had been castout, its data entry might not have been reclaimed.

Alternatively, the page might have been written to the group buffer pool because GBPCACHE ALL had been specified. In either of these cases, the page would be available in the group buffer pool when needed.

2.5: Castout Process

Changed pages do not stay in the group buffer pool indefinitely. Eventually, the pages are written to DASD.

DB2 uses a *castout* process to write **changed**³ data from a group buffer pool, through a DB2 private buffer, to DASD. The castout process occurs during a group buffer pool checkpoint. Between checkpoints, castout normally is controlled by the value specified for two group buffer pool thresholds:

- **Class Castout Threshold.** The Class Castout Threshold is a percent of changed pages on a castout class queue. This threshold determines the total number of *changed* pages that can exist in a castout class queue before castout occurs.

There are a fixed number of castout class queues in each group buffer pool. DB2 internally maps *changed* pages that belong to the same page sets or partitions to the same castout class queues in the group buffer pool. Since there are a fixed number of queues, more than one page set or partition can be mapped to a single castout class queue.

When DB2 writes changed pages to the group buffer pool, it determines how many *changed* pages are on a particular class castout queue. If the Class Castout Threshold (changed pages on queue divided by total pages in group buffer pool) is exceeded for any queue, DB2 casts out a number of pages from that queue to bring the percent of changed pages below the Class Castout Threshold.

The Class Castout Threshold is conceptually similar to the VDWQT threshold for local buffer pools, except that the castout class queue may contain multiple page sets or partitions.

The Class Castout Threshold has a default of 10, meaning that castout begins when changed pages on a castout class queue are more than 10% of the total pages in the group buffer pool. This default can be changed by the ALTER GROUPBUFFERPOOL CLASST command.

- **Group Buffer Pool Castout Threshold.** The Group Buffer Pool Castout Threshold is a percentage of the total number of pages in the group buffer pool. This threshold determines the total number of *changed* pages that can exist in the group buffer pool before castout occurs.

³Unchanged pages in the group buffer pool are not castout. Unchanged pages would be in the group buffer pool if the GBPCACHE ALL option were used.

When DB2 writes changed pages to the group buffer pool, it determines how many changed pages are in the group buffer pool. If the Group Buffer Pool Castout Threshold (changed pages divided by total pages in group buffer pool) is exceeded, DB2 casts out pages from class castout queues to bring the percent of changed pages below the Threshold.

The Group Buffer Pool Castout Threshold is conceptually similar to the DWQT threshold for local buffer pools.

The Group Buffer Pool Castout Threshold has a default of 50, meaning that castout begins when changed pages in the group buffer pool are more than 50% of the total pages in the group buffer pool. This default can be changed by the ALTER GROUPBUFFERPOOL GBPOOLT command.

The actual castout is done by the “owner” of the page set or partition. The DB2 subsystem that is assigned “ownership” of castout for a page set or partition is the DB2 subsystem that had the first *update intent* on the page set or partition. After the castout ownership is assigned, subsequent updating DB2 subsystems become backup owners. One of the backup owners becomes the castout owner if the original castout owner should no longer have R/W interest in the page set.

Exhibit 6 illustrates the castout process.

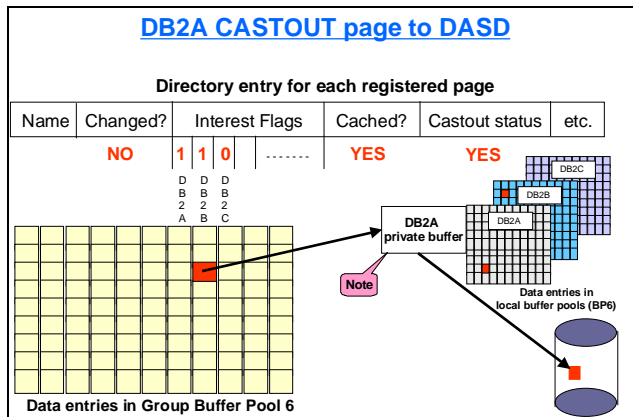


Exhibit 6: Castout process

Even after the page has been castout, **it normally remains in the group buffer pool**, and can be read by any DB2 member. However, the status of a castout page is altered from “changed” to “unchanged” since it has been written to DASD. Consequently, the data entry of the castout page can be “reclaimed” by DB2 if necessary.

This feature (leaving castout pages in the group buffer pool) is conceptually similar to that with local buffer pools (changed pages in a local buffer pool remain in the pool

even after they have been written to DASD or to a group buffer pool).

2.6: Reclaiming Entries

A group buffer pool has a fixed number of directory entries and data entries. Directory entries are used when some DB2 data sharing member registers a page with the group buffer pool. Data entries are used when a changed page is written to the group buffer pool (with GBPCACHE CHANGED), or when an unchanged page is written to the group buffer pool (with GBPCACHE ALL).

Directory entries are released when there is no more R/W interest in a page set or partition. Data entries are not “released” as such. Their status changes from “changed” to “unchanged” when they are castout. Additionally, the storage occupied by data entries is simply available when the corresponding directory entry is released.

However, the amount of allocated group buffer pool storage can be inadequate for the DB2 data sharing workload. In such a situation, DB2 must “reclaim” data entries or directory entries.

- **Reclaiming data entries.** Data entries for changed pages are “reclaimed” by the castout process described in Section 2.5 of this document. As described in that section, castout pages simply have their status altered from “changed” to “unchanged” in their associated directory entry once they are castout.

Unchanged pages in the group buffer pool can be reclaimed by DB2 when space is needed to write a page to the group buffer pool, but there is no unused data entry storage. In this case, DB2 reclaims “unchanged” pages on a “least recently used” (LRU) basis.

- **Reclaiming directory entries.** Directory entries can be reclaimed when a DB2 data sharing member wishes to register a page as a part of reading the page, but there are no directory entries available.

When directory entries are reclaimed, the associated pages must be invalidated in each DB2 registered for the pages. When a page has been invalidated, any DB2 member of the data sharing group must reread the page from DASD if the DB2 member again wishes to reference the page. Rereading the page requires re-registration of the page (including creating a directory entry). This can cause considerable overhead and delay to the application.

The performance implications of reclaiming data entries and directory entries are explored in the next section.

3: ANALYZING DB2 DATA SHARING

DB2 data sharing has many laudatory features, as mentioned at the beginning of this document.

However, some users complain that performance is poor with DB2 data sharing, or users complain that there is excessive overhead.

Often, these complaints are valid only because of improper sizing of group buffer pools, an inappropriate directory entry/data entry ratio for a group buffer pool when considering the specific DB2 workload, incorrect DB2 data sharing parameters, or even inappropriate parameters for DB2 virtual buffer pools that adversely affect DB2 data sharing performance.

This section describes standard data that is available for analyzing DB2 data sharing performance problems, discusses some of the common problems that occur with DB2 data sharing, describes the analysis that is used to identify the problems, and suggests alternatives that should be considered to solve the problems.

3.1: DB2 Performance Data

The data that normally is used to analyze DB2 data sharing performance problems come from standard System Management Facility (SMF) records.

- **DB2 interval statistics** are collected continuously while DB2 is running, and the statistics are recorded to SMF at regular intervals. The default recording interval is 30 minutes. Recording at 15 minutes adds little overhead, but can significantly enhance the analysis of DB2 performance problems. The DB2 interval statistics are written to SMF Type 100.

The interval statistics contain group buffer pool information that can be analyzed to determine how well DB2 data sharing is performing, and to determine causes of poor performance.

The interval statistics also contain *virtual buffer pool statistics*. It often is necessary to analyze the virtual buffer pool information in order to determine whether local buffer pools are causing problems for group buffer pools.

- The **group buffer pool attributes** records are written when statistics trace class 5 is on (also written if monitor class 1 is on). These records contain information such as the size of the group buffer pools, castout thresholds, etc. Since the group buffer pool parameters can be dynamically changed (via ALTER GROUPBUFFERPOOL), change information is important when analyzing performance data.

- The **virtual buffer pool attributes** records are written at each statistics interval (also written if monitor trace class 1 is on, and are written each time IFCID 106 is written because of a start trace command). These records contain information such as the size of the local buffer pools, deferred write thresholds, etc. Since the virtual buffer pool parameters can be dynamically changed (via ALTER BUFFERPOOL), change information is important when analyzing performance data.
- **SMF Type 74 (Subtype 2 and Subtype 4)** records contain information about XCF activity and coupling facility activity, respectively. This information is very useful to help identify problems with coupling facility requests and delays. Information is provided for both the DB2 lock structure and the DB2 group buffer pools, from the view of XCF. The structure names identify whether a structure is a DB2 lock or group buffer pool structure.
- **DB2 accounting information** is written to SMF Type 101 records. This source of performance analysis data is listed last because the records are voluminous and require significant processing time. The accounting information would be analyzed only if it were necessary to associate performance problems to specific applications.
- **Data set information** is available in SMF Type 42 records. This information would be analyzed to identify data set problems.

3.2: Common DB2 Data Sharing Problems

Analyzing standard data related to DB2 performance can usually identify the cause of problems, and changes normally can be made to significantly improve performance of DB2 data sharing.

The examples given in the following sections are meant to illustrate the analysis and possible solutions. Neither the analysis nor the solutions are complete; other areas can be analyzed and other solutions can be explored in each area of analysis.

In some instances, conflicts between applications might be so great that it might be impossible to obtain satisfactory DB2 data sharing performance. In this case, it might be necessary to decrease the data sharing level associated with specific group buffer pools. To accomplish this, page sets or partitions could be reassigned to different virtual buffer pools (with corresponding reassignment to different group buffer pools), additional virtual buffer pools could be created (with corresponding additional group buffer pools), or some application could be moved between data sharing members and moved to another processing time.

3.2.1: Coupling facility read requests could not complete

As mentioned in the first part of this document, a page must be registered with the group buffer pool before it is read from DASD. If a directory entry exists for the page, the directory entry is updated to reflect that *this* DB2 has an interest in the page. If a directory entry does not exist for the page, one is created and then updated to reflect that *this* DB2 has an interest in the page.

Creating a directory entry means that storage must be available in the group buffer pool to hold the directory entry. If many pages are registered (or if the ratio of directory entries to data entries is too low), there might be no storage in the group buffer pool that is available for creating a directory entry. In this case, the DASD read fails and directory entries must be **reclaimed** in the group buffer pool so a new directory entry can be created for the page. This situation can cause serious performance problems:

- When a DASD read fails, the DB2 application must wait for the page if the read were a synchronous read (a random page read or a sequential page read that had not been read as part of a prefetch operation).
- When a DASD read fails for a prefetch operation, the prefetch operation is aborted. This means that pages that otherwise would be prefetched via an asynchronous I/O operation must be read via a synchronous I/O operation when the pages are required by the application.
- When directory entries are reclaimed, the associated pages must be invalidated in each DB2 registered for the pages. This page invalidation process creates overhead and uses system resources.
- When a page has been invalidated in each DB2 registered for the page, those DB2s must reread the page from DASD if the DB2s again wish to reference the page. Rereading the page requires re-registration of the page (including creating a directory entry). Reclaiming pages because of a shortage of directory entry storage can cause “thrashing” of pages between buffer pools and DASD, and can generate significant overhead.

This problem is detected by examining the QBGLRF (read fail) variable in the DB2 interval statistics. A problem is indicated when the QBGLRF value is non-zero.

Possible solutions to this problem include the following alternatives:

- **Increase the directory entry/data entry ratio.** The easiest solution to this problem is to simply increase

the directory entry/data entry ratio. This can be done by ALTER GROUPBUFFERPOOL.

Recall that the directory entry size is small relative to the size of the data entry (directory entries are normally about 200 bytes while data entries are the size of the local buffer pool page, which is a minimum of 4096 bytes). Consequently, increasing the directory entry/data entry ratio might yield a relatively large increase in the number of directory entries, while not dramatically reducing the number of data entries.

- **Increase the size of the group buffer pool.** If the above solution is not feasible because of a relative shortage of data entries, consider increasing the size of the group buffer pool. Increasing the size of the group buffer pool will, of course yield more directory entries (since part of the increase would be used for directory entries and part would be used for data entries). If the size of the group buffer pool is increased, consider increasing the directory entry/data entry ratio at the same time. This would cause the increased group buffer pool storage to be used for directory entries.
- **Reduce the data sharing level of the group buffer pool.** If neither of the above alternatives is feasible, consider reducing the data sharing level of the group buffer pool. The beginning of this section describes some ways to reduce the data sharing level associated with a specific group buffer pool.

3.2.2: Coupling facility write requests could not complete

A data entry must be available in the group buffer pool when a page of data is to be written to the group buffer pool. If the group buffer pool does not have a data entry available, the write is rejected because of a lack of storage. When a write is rejected, DB2 initiates castout processing if castout is not already in progress. The DB2 member waits for 3 seconds after triggering the castout, and it retries the write operation for up to four times. Pages that cannot be written to the group buffer pool are added to the Logical Page List and message DSNB250E is issued. This action would require a recovery of the page set before the pages could be used.

This problem is detected by examining the QBGLWF (write fail) variable in the DB2 interval statistics. A problem is indicated when the QBGLWF value is non-zero.

Possible solutions to this problem include the following alternatives:

- **Reduce the local buffer pool thresholds.** Failed writes typically result from castout processing not being

able to keep up with the writes. There are several causes, but the most common is that a transaction commits and “floods” the group buffer pool with write requests. This “flood” of write requests can be reduced by reducing the local buffer pool thresholds (VDWQT and DWQT). This would spread the write requests more evenly over the life of the transaction.

- **Increase frequency of castout.** By increasing the frequency of castout processing, pages in the group buffer pool would be castout to DASD more often. Consequently, more pages would normally be available. This is accomplished by reducing the group buffer pool checkpoint interval (the default is eight minutes), reducing the group buffer pool castout threshold, or reducing the group buffer pool class castout threshold.
- **Increase the size of the group buffer pool.** As mentioned above, the basic cause of the failed write is that a data entry was not available. Consider increasing the size of the group buffer pool if the above actions do not yield satisfactory results.
- **Issue COMMIT more frequently.** If the application can issue COMMIT more frequently, few pages would normally be written to the group buffer pool at each COMMIT. Consequently, there would be a greater opportunity for the castout processing to castout pages between COMMIT operations.
- **Verify GBPCACHE ALL is required.** Some applications might have specified GBPCACHE ALL, and DB2 could be “flooding” the group buffer pool during periods of heavy prefetch activity. In this case, investigate these applications to determine whether this specification is necessary.
- **Review I/O configuration.** Check the I/O configuration (and the coupling facility performance) to ensure that castout processing is not being delayed by I/O waits.
- **Reduce the directory entry/data entry ratio.** It is possible that the directory entry/data entry ratio is too high, resulting in too much storage reserved for directory entries rather than being available for data entries.

However, the directory entry is small relative to the size of the data entry (directory entries are normally about 200 bytes while data entries are the size of the local buffer pool page, which is a minimum of 4096 bytes). Consequently, adjusting the directory entry/data entry ratio usually would have little positive effect in this situation.

3.2.3: Low read hit percentage for cross-invalidated pages

When a DB2 data sharing member writes changed pages to the group buffer pool, the page is invalidated in the buffer of all other DB2 members. A major reason for writing the page to the group buffer pool is the expectation that other data sharing members will eventually read the changed page, rather than having to read the page from DASD. If the changed page is not read again, there is considerable overhead required to write the page to the group buffer pool, overhead for cross-invalidation, overhead for castout processing, and overhead for re-registering the page. This overhead is a waste of system resources.

More significant is the situation where the invalidated page is required by another data sharing member, but the page has already been castout to DASD, and its data entry has been reclaimed for another page. In this case, not only is the overhead a waste of system resources, but the DASD accesses (both for castout and for reread by the other data sharing member) could potentially be avoided.

The situation is caused by the “residency time” of the pages being too short before the pages are castout.

This problem is detected by computing the read hit percent for cross-invalidated pages:

$$\text{Read hit percent} = \frac{QBGLXD}{QBGLXD + QBGLXR} * 100$$

Based on IBM's guidance, a problem is indicated when the read hit percent for cross-invalidated pages is less than 95%.

Possible solutions to this problem include the following alternatives:

- **Increase Class Castout Threshold or Group Buffer Pool Castout Threshold.** Pages normally are castout when a group buffer pool checkpoint occurs, or when the Class Castout Threshold or Group Buffer Pool Castout Threshold is reached. If only a few page sets or partitions are assigned to the group buffer pool, castout might occur prematurely because of the castout thresholds. If a low total read hit percentage occurs, consider increasing the thresholds.

However, since all changed pages are castout at group buffer pool checkpoint, The “castout engine not available” situation might be encountered if the thresholds are excessively high. This situation is described in Section 3.2.4.

- **Verify that GBPCACHE ALL is required.** Some application may have specified GBPCACHE ALL and DB2 was writing pages to the group buffer pool regardless of whether they were changed. In this case, review the total read hit percentage (changed and unchanged pages) to see if GBPCACHE ALL is productive.
- **Consider GBPCACHE NONE.** One or more of the page sets assigned to the group buffer pool might be a candidate for GBPCACHE NONE (if operating with DB2 Version 6)⁴. This alternative would be considered, of course, only if total read hit percentage was very low (e.g., the percentage was less than 5%).
- **Increase the size of the group buffer pool.** As mentioned above, the basic cause of the low read hit percentage for cross-invalidate pages is that the “residency time” of the pages is too short before the pages are castout. Consider increasing the size of the group buffer pool if the above actions do not yield satisfactory results.
- **Review I/O configuration.** Another common reason that a castout engine is not available is that the castout rate occurs too quickly for the coupling facility and the DASD system to handle the I/O, or that the DB2 “owning” the group buffer pool (and responsible for castout processing) is overloaded with higher importance work. Verify that the coupling facility and the DASD I/O system are not experiencing unnecessary delay, and that sufficient capacity exists for the DB2 that owns the group buffer pool.
- **Reduce the group buffer pool castout thresholds.** The Class Castout Threshold or the Group Buffer Pool Threshold might be too high, with the result that a burst of write activity to the group buffer pool could cause a corresponding burst of castout activity. Decreasing these thresholds could reduce the bursts of castout.
- **Increase the size of the group buffer pool.** Increasing the size of the group buffer pool might help this situation, but only if the bursts of activity described above can be managed.

3.2.4: Castout engine was not available

When a castout operation is scheduled, a *castout engine* is used for the castout process. DB2 has up to 300 castout engines, depending on the castout workload and the amount of DBM1 storage available. When castout is scheduled, there might not be an available engine. In this situation, castout cannot commence for the castout process scheduled, and castout waits for an engine to become available.

This problem is detected by examining the QBGLCN (castout engine not available) variable in the DB2 interval statistics. A problem is indicated when the QBGLCN value is non-zero.

Possible solutions to this problem include the following alternatives:

- **Reduce the local buffer pool thresholds.** Local buffer pool write activity can occur in bursts when a transaction commits and “floods” the group buffer pool with write requests for changed pages. This “flood” of write requests can cause the group buffer pool thresholds to be repeatedly reached, with corresponding castout operations initiated. A “flood” of write requests can be reduced by lowering the local buffer pool thresholds (VDWQT or DWQT). This will spread the write requests more evenly over the life of the transaction.

⁴If GBPCACHE NONE is specified, IBM suggests that DASD Fast Write should be enabled. Additionally, the vertical deferred write threshold (VDWQT) should be set to 0, to cause deferred write pages to be continuously written. Continuously writing these pages would avoid a large surge of write activity at COMMIT.

3.2.5: Coupling facility write engine was not available

The coupling facility write operations are implemented by DB2 subtasks, called *write engines*. The maximum number of concurrent write subtasks is a constant designed into DB2 (the maximum *write engines*). Coupling facility write operations are suspended if there are no more write engines when DB2 wishes to write pages to the coupling facility.

When a “must complete” operation to the group buffer pool fails (a write at commit or rollback, or a read for rollback or restart), the page is put in the *logical page list* (LPL) and becomes unavailable. The START DATABASE command must be used to recover pages from the LPL.

This problem is detected by examining the QBGLSU (coupling facility write engine not available) variable in the DB2 interval statistics. A problem is indicated when the QBGLSU value is non-zero.

Possible solutions to this problem include the following alternatives:

- **Reduce the local buffer pool thresholds.** Local buffer pool write activity can occur in bursts when a transaction commits and “floods” the group buffer pool with write requests for changed pages. This “flood” of write requests can cause the pool of write engines to be exhausted. The “flood” of write requests can be reduced by lowering the local buffer pool thresholds (DWQT or VDWQT). Lowering the thresholds would

spread the write requests more evenly over the life of the transaction.

- **Verify GBPCACHE ALL is required.** Some application may have specified GBPCACHE ALL and DB2 was writing pages to the group buffer pool regardless of whether they were changed. In this case, review the total read hit percentage (changed and unchanged pages) to see if GBPCACHE ALL is productive.
- **Issue COMMIT more frequently.** If the application can issue COMMIT more frequently, few pages would normally be written to the group buffer pool at each COMMIT. This would minimize the number of changed pages written from the local buffer pool when the application does issue COMMIT.

3.2.6: Excessive cross-invalidations due to directory reclaims

When directory entries are reclaimed to handle new work, cross-invalidation must occur for all members of the DB2 data sharing group that have those pages in their buffer pools, even when the data has not actually changed. This cross-invalidation process requires overhead, as all members must be notified that their copy of the page is invalid.

If another member of the DB2 data sharing group needs access to the invalidated page, that DB2 must read the page from DASD. This means that the DB2 must first register the page with the coupling facility (it must acquire a directory entry and register interest in the page), and then read the page from DASD.

Directory reclaim activity can result in (1) increased read I/O activity to the data base to reacquire a referenced data item, (2) increased CPU utilization and coupling facility overhead caused by re-registering interest in pages, and (3) elongated transaction response times. In some situations, the directory reclaim activity and cross-invalidation can lead to "thrashing" among DB2 members, with significant overhead and poor DB2 performance.

This problem is detected by computing the percent cross-invalidated pages due to directory reclaims:

$$\text{Percent XI (directory reclaims)} = \frac{R744CXDR}{R744CXDR + R744CDER} * 100$$

Based on IBM's guidance, a problem is indicated when the read hit percent for cross-invalidated pages is greater than zero (i.e., when there are any cross-invalidations due to directory reclaims).

Possible solutions to this problem include the following alternatives:

- **Increase the directory entry/data entry ratio.** Excessive cross-invalidations due to directory reclaims indicate that there are too few directory entries. The number of directory entries can be increased by increasing the directory entry to data entry ratio. Implementing this option would reduce the number of data entries, but changing the directory entry to data entry ratio is relatively easy to implement. A small increase in the directory entry to page entry ratio could result in a large increase in the number of directory entries, with only a small decrease in the number of data entries. This is because the directory entry is so small (about 200 bytes) relative to the data entry (4096 bytes or more, depending on the page size of the corresponding local buffer pool).
- **Increase the size of the group buffer pool.** Since the group buffer pool is divided into directory entries and data entries, increasing the size of the group buffer pool would provide more directory entries. This option is not likely to be satisfactory, however, unless either (1) the size of the group buffer pool is significantly increased, or (2) the directory entry to data entry ratio is increased.

4: CONCLUSIONS

This document has described some of the more serious problems with DB2 data sharing. Fortunately, these problems are easy to detect from standard data that is available in SMF records. Many of the listed alternatives are easy to implement, yet the solutions can result in significantly improved performance with DB2 data sharing.

REFERENCES

"Parallel Sysplex Tuning Update" series, Joan Kelley, IBM, **Session 2523 at SHARE Winter 2001 in Long Beach.**

"Coupling Facility and DB2 data sharing dance to the same tune," Linda August, IBM, **Session 2524 at SHARE Winter 2001 in Long Beach.**

DB2 Data Sharing: Planning and Administration, DB2 Version 4, SC26-3269

DB2 Data Sharing: Planning and Administration, DB2 Version 5, SC26-8961

DB2 Data Sharing: Planning and Administration, DB2 Version 6, SC26-9007

DB2 Universal Database for OS/390 and z/OS: Administrative Guide, DB2 Version 7, SC26-9931

DB2 Administration Guide, Version 4, SC26-3269

DB2 Administration Guide, Version 5, SC26-8957

DB2 Administration Guide, Version 6, SC26-9003

DB2 Universal Database for OS/390 and z/OS: Data Sharing and Administration, DB2 Version 7, SC26-9935

DB2 for MVS/ESA Version 4 Data Sharing Performance Topics Redbook, SG24-4611

DB2 for OS/390 Version 5 Performance Topics Redbook SG24-2213

DB2 UDB for OS/390 Version 6 Performance Topics Redbook, SG24-5351

OS/390 MVS Setting Up a Sysplex. GC28-1779

OS/390 MVS Parallel Sysplex Configuration: Volume 2 Cookbook, SG24-2076

DSNWMSGs (IFCD field descriptions) for DB2 UDB for OS/390 Version 6