Rule DB2-235: Dynamic sequential prefetch was invoked frequently

Finding: DB2 invoked dynamic sequential prefetch a large percent of total sequential prefetch, as a result of sequential detection.

Impact: This finding can have a LOW IMPACT or MEDIUM IMPACT on the performance of the DB2 subsystem.

This finding is suppressed beginning with DB2 UDB for z/OS Version 8.

Discussion: Before DB2 UDB for z/OS Version 8.1, DB2 uses three read mechanisms: normal read, sequential prefetch, and list sequential prefetch.

- **Normal Read**: Normal read is used when just one or a few consecutive pages are retrieved. The unit of transfer for a normal read is one page.

  Additionally, normal read is performed when the number of concurrent sequential prefetch I/O operations exceeds the number of DB2 sequential prefetch *read engines*. The maximum number of sequential prefetch read engines is a constant within DB2 and is set to 300. When the maximum read engines are active, DB2 disables sequential prefetch and performs asynchronous I/O operations. Please refer to Rule DB2-231 for additional information.

- **Sequential Prefetch**: Sequential prefetch is a mechanism that triggers consecutive asynchronous I/O operations. Sequential prefetch brings pages into the virtual buffer pool before they are required and reads several pages with a single I/O operation. Consequently, sequential prefetch allows CPU processing and I/O operations to be overlapped.

  Sequential prefetch can be used to read data pages, by table space scans, or index scans with clustered data reference. It can also be used to read index pages in an index scan.

- **List Sequential Prefetch**: List Sequential Prefetch is used to prefetch data pages that are not contiguous (such as through non-clustered indexes). List prefetch can also be used by incremental image copy.

With DB2 UDB for z/OS Version 8, DB2 also uses **Dynamic prefetch**. When using dynamic prefetch, DB2 can automatically adjust between multi-page prefetch and single page synchronous read, as needed for optimal I/O performance. Dynamic prefetch can reduce paging and improve performance over sequential prefetch, especially for access to data that might be arranged sequentially for some sets of pages but scattered randomly on other pages.

With DB2 UDB for z/OS Version 9, DB2 uses dynamic prefetch in most situations, with a few exception (such as table space scans).

The following discussion applies only with DB2 prior to DB2 UDB for z/OS Version 8.

DB2 normally selects sequential prefetch at bind time. If DB2 does not choose prefetch at bind time, it can sometimes use prefetch at execution time by a method called sequential detection, or dynamic sequential prefetch.

DB2 can use sequential detection for both index leaf pages and data pages. It is most commonly used on the inner table of a nested loop join, if the data is accessed sequentially.

If a table is accessed repeatedly using the same statement (for example, DELETE in a do-while loop), the data or index leaf pages of the table can be accessed sequentially. This is common in a batch processing environment. Sequential detection can then be used if access is through:

- SELECT or FETCH statements
- UPDATE and DELETE statements
- INSERT statements when existing data pages are accessed sequentially

The pattern of accessing a page is tracked when the application scans DB2 data through an index. Tracking is done to detect situations where the access pattern that develops is sequential or nearly sequential.

The most recent 8 pages are tracked. A page is considered page-sequential if it is within P/2 advancing pages of the current page, where P is the prefetch quantity. P is usually 32.

If a page is page-sequential, DB2 determines further if data access is sequential or nearly sequential. Data access is declared sequential if more than 4 out of the last 8 pages are page-sequential; this is also true for index-only access. The tracking is continuous, allowing access to slip into and out of data access sequential mode.

DB2 can use sequential detection if it did not choose sequential prefetch at bind time because of an inaccurate estimate of the number of pages to be accessed. This is because at bind time, the number of pages to be accessed can only be estimated. Consequently, sequential detection acts as a safety net and is employed when the data is being accessed sequentially.

Implementing dynamic sequential prefetch using sequential detection can significantly improve performance over asynchronous reads. However, sequential detection results in increased DB2 overhead. This overhead should be eliminated if possible.

Additionally, dynamic sequential prefetch is not as efficient as normal sequential prefetch selected at bind time, with respect to I/O buffer handling.

CPExpert uses the QBSTDPF variable in DB2STATB (the number of dynamic sequential prefetch requests) and the QBSTSEQ variable (the number of sequential prefetch requests, excluding dynamic sequential prefetch requests) to compute the percent of total sequential prefetch requests that were dynamic sequential prefetch requests. To avoid spurious reporting, CPExpert ignores any interval in which dynamic sequential prefetch occurred less than 1,000 times in the interval.

CPExpert computes the percent of total sequential prefetch operations that were as a result of sequential detection. The calculation uses data in DB2STATB, and is performed as shown below:

$$\textit{Percent dynamic sequential prefetch} = \frac{\textit{Dynamic sequential prefetch}}{\textit{Dynamic sequential prefetch} + \textit{Sequential prefetch}}$$

CPExpert produces Rule DB2-235 when the percent of requests that were dynamic sequential prefetch requests exceeds the value specified by the **PCTDPF** guidance variable.

The default value for the **PCTDPF** guidance variable is 0, indicating that CPExpert should produce Rule DB2-235 whenever dynamic sequential prefetch was implemented. **Please note that this low threshold was selected only to alert you to the performance issue. You may wish to adjust the threshold for a particular buffer pool after you considered the alternatives listed below.** Also, please recall that CPExpert will ignore any interval in which less than 1,000 dynamic sequential prefetch operations occurred.

The following example illustrates the output from Rule DB2-235:

```
RULE DB2-235: DYNAMIC SEQUENTIAL PREFETCH WAS HIGH

   Buffer Pool 3: DB2 invoked dynamic sequential prefetch a large percent
   of total sequential prefetch, as a result of DB2's sequential detection.
   Please review the discussion in the DB2 Component User Manual related
   to this finding.  You should either use the RUNSTATS utility to reduce
   the percent of dynamic detection, or use the PCTDPF guidance variable
   to alter CPExpert's analysis of dynamic sequential prefetch for this
   buffer pool. This situation occurred for Buffer Pool 3 during the
   intervals shown below:

                             --SEQUENTIAL PREFETCH OPERATIONS--   PERCENT
   MEASUREMENT INTERVAL      NORMAL        DYNAMIC        TOTAL   DYNAMIC
    0:53- 1:23, 15SEP1999     1,966          6,868        8,834     77.7
    1:23- 1:53, 15SEP1999     4,701          7,288       11,989     60.8
    4:52- 5:22, 15SEP1999     3,087          1,140        4,227     27.0
    7:22- 7:51, 15SEP1999    10,151          1,008       11,159      9.0
    8:21- 8:51, 15SEP1999     6,428          6,185       12,613     49.0
    8:51- 9:21, 15SEP1999     2,375          1,927        4,302     44.8
    9:21- 9:51, 15SEP1999     2,660         10,720       13,380     80.1
    9:51-10:21, 15SEP1999     2,578          2,339        4,917     47.6
   10:21-10:51, 15SEP1999    19,075          3,151       22,226     14.2
```

Suggestion: If Rule DB2-235 is produced more than occasionally, you should consider the following alternatives:

- Use the RUNSTATS utility so that the DB2 catalog statistics take into account the newly loaded data, and SQL paths can be selected with accurate information. Following this, rebind any application plans that depend on the loaded tables. This is necessary to update the path selection of any embedded SQL statements.

- Review the characteristics of the DB2 activity, since it is possible that DB2 will not select sequential prefetch regardless of the RUNSTATS values. For example, a single SQL SELECT statement is not usually optimized with sequential prefetch I/O at bind time. Additionally, ad hoc queries would not be bound, and the DB2 optimizer would not select sequential prefetch. *Dynamic sequential detection would be the most optimum approach in this environment.*

- You can alter CPExpert’s analysis by modifying the **QBSTDPF** guidance variable in USOURCE(DB2GUIDE).

Reference: DB2 for OS/390 Version 3: Administration Guide
Section 7.9.5.2 (Sequential Detection at Execution Time)

DB2 for OS/390 Version 4: Administration Guide
Section 5.10.5.3 (Sequential Detection at Execution Time)

DB2 for OS/390 Version 5: Administration Guide
Section 5.10.5.3 (Sequential Detection at Execution Time)

DB2 for OS/390 Version 6: Administration Guide
Section 5.10.5.3 (Sequential Detection at Execution Time)

DB2 UDB for OS/390 and z/OS, Version 7: Administration Guide
Section 5.10.5.3 (Sequential Detection at Execution Time)

DB2 UDB for z/OS, Version 8: Administration Guide |
Chapter 33. Using EXPLAIN to improve SQL performance |
Sequential Detection at Execution Time |