## Rule CIC277: CICS-DB2 pool threads in use was approaching thread limit

**Finding:** The number of pool threads in use was approaching the thread limit specified for the CICS-DB2 connection.

**Impact:** This finding does not have an impact on the performance of CICS tasks in the region that use the CICS-DB2 connection. However, the finding provides an "early warning" of a situation that could have a MEDIUM IMPACT or HIGH IMPACT on the performance of CICS tasks in the region that use the CICS-DB2 connection.

**Logic flow:** This is a basic finding, based upon an analysis of the CICS statistics. This finding applies only with CICS/Transaction Server for OS/390 Release 1.2 and subsequent releases of CICS.

**Discussion:** The CICS DB2 attachment facility creates an overall connection between CICS and DB2. CICS applications use this connection to issue commands and requests to DB2.

A CICS transaction accesses DB2 via a *thread*, which is an individual connection into DB2. Threads are created when they are needed by transactions, at the point when the application issues its first SQL or command request. The transaction uses the thread to access resources managed by DB2.

There are three types of threads: Command threads, Pool threads, and DB2ENTRY threads,.

- Command threads are used by the CICS DB2 attachment facility for issuing commands to DB2 via the DSNC transaction.

- Pool threads are used for all transactions and commands that are not using a Command thread (because the transaction is not DSNC), are not using an Entry thread (because an Entry thread had not been defined for the transaction), or have been "overflowed" to the pool because a Command thread or an Entry thread was not available.

- One or more Entry thread categories optionally can be defined (using the DB2ENTRY definition) for specific transactions or groups of transactions. Entry threads are used for transactions that need to be managed separately from the normal transactions, or for transactions that have special accounting needs.

©Copyright 1994, Computer Management Sciences, Inc. 

When a thread is no longer needed by the transaction, the thread is released. The thread release typically occurs after syncpoint completion. The thread may be terminated immediately upon release or it may be retained for a period of time, depending on the type of thread and whether "thread protection" has been specified .

Transactions can use Pool threads in one of three ways:

- If the transaction is not a DSNC transaction and is not assigned to an Entry thread, the transaction automatically attempts to use a Pool thread.

- If a DSNC transaction is submitted and no command threads are available, the transaction automatically "overflows" to use a Pool thread.

- If a transaction is assigned to an Entry thread, but no Entry threads are available, the transaction can either wait, can abend, or can "overflow" to use a Pool thread (the THREADWAIT attribute for the DB2ENTRY definition controls the action that should be taken).

Regardless of how transactions use pool threads, a pool thread must be available when the transaction attempts acquire a thread. The number of Pool threads that can be available is controlled by the THREADLIMIT attribute for the DB2CONN definition. Specifying an optimal value for the THREADLIMIT attribute is a balance between specifying a value that is too large for the environment (and wasting storage) or specifying a value that is too small for the environment (and incurring either transaction abends or transaction waits).

If a pool thread is not available, the transaction is either abended or placed on a Ready Queue, waiting for a pool thread to become available (the THREADWAIT attribute for the DB2CONN definition controls the action that should be taken).

With Rule CIC277, CPExpert provides an "early warning" of a potential performance problem with the number of threads in use approaching the limit set by the THREADLIMIT attribute.

CICS-DB2 global statistics are available in MXG file CICDB2GL. CPExpert uses data in CICDB2GL to calculate the percent of pool threads in use relative to the Pool Thread Limit, using the following algorithm:

$$\textit{Percent pool threads in use} = \frac{\textit{Peak pool threads in use}}{\textit{Pool Thread Limit}}$$

©Copyright 1994, Computer Management Sciences, Inc. 

where

| | | |
|---|---|---|
| Peak pool threads in use | = | D2GTHRPK |
| Pool Thread Limit | = | D2GTHRLM |

CPExpert produces Rule CIC277 when the percent pool threads in use is more than the value specified by the **PCTD2THR** guidance variable in USOURCE(CICGUIDE). The default value for the **PCTD2THR** is 80 indicating that CPExpert should produce Rule CIC277 whenever the pool threads in use was more than 80% of the pool thread limit specified for the CICS-DB2 Connection.

**Suggestion:** If Rule CIC276 is produced frequently, you should consider the following alternatives:

- **Increase THREADLIMIT value**. It normally is not wise to have more than one transaction on the Pool Ready Queue, since this indicates that there is a delay to transaction response caused by internal queuing. You can increase the THREADLIMIT value on the DB2CONN definition if you wish to allow more pool threads to be used.

- **Use transaction class limits**. If you wish to limit the amount of CISC-DB2 activity, you should consider using transaction class limits rather than using the THREADLIMIT value. IBM states that it is better to limit transactions using a transaction class than allow them to queue for threads.

- **Modify guidance**. You can modify the PCTD2THR guidance variable in USOURCE(CICGUIDE) if you feel that Rule CIC277 is produced prematurely.

**Reference**: *CICS/TS Release 1.3 CICS DB2 Guide*: Section 5.4 (Creating, using, and terminating threads)

CICS/TS Release 1.3 *Resource Definition Guide*: Section 5.1.3 (DB2CONN)

CICS/TS for z/OS Release 2.2 *CICS DB2 Guide*: Section 5.4 (How threads are created, used, and terminated)

CICS/TS for z/OS Release 2.2 *Resource Definition Guide*: Section 2.3.4 (DB2 connection definition attributes)

**Thanks**: Thanks to **Rex Avendano** (Kaiser Permenante) for suggesting this rule.

©Copyright 1994, Computer Management Sciences, Inc. 