DB2 magazine

http://www.db2mag.com/db_area/archives/1998/q1/98spDba.shtml

Tuning Your RCT

How to exploit CICS Resource Control Table options for cost and performance.

By David Beulke

Spring 1998

Printer-Friendly Version Email this Story Bookmark to del.ico.us Digg It!

As DB2 applications are developed in the CICS environment, application response time is dependent on a little, often neglected, assembler module: the CICS Resource Control Table (RCT). This RCT module -- like the other CICS environment tables -- has many parameters and options. These parameters are especially important when preparing for DB2 data sharing to maximize CICS thread reuse, minimize thread termination costs, and avoid locking. In this column I'll discuss how you can exploit the RCT tuning options to improve your DB2 CICS transaction performance and response time.

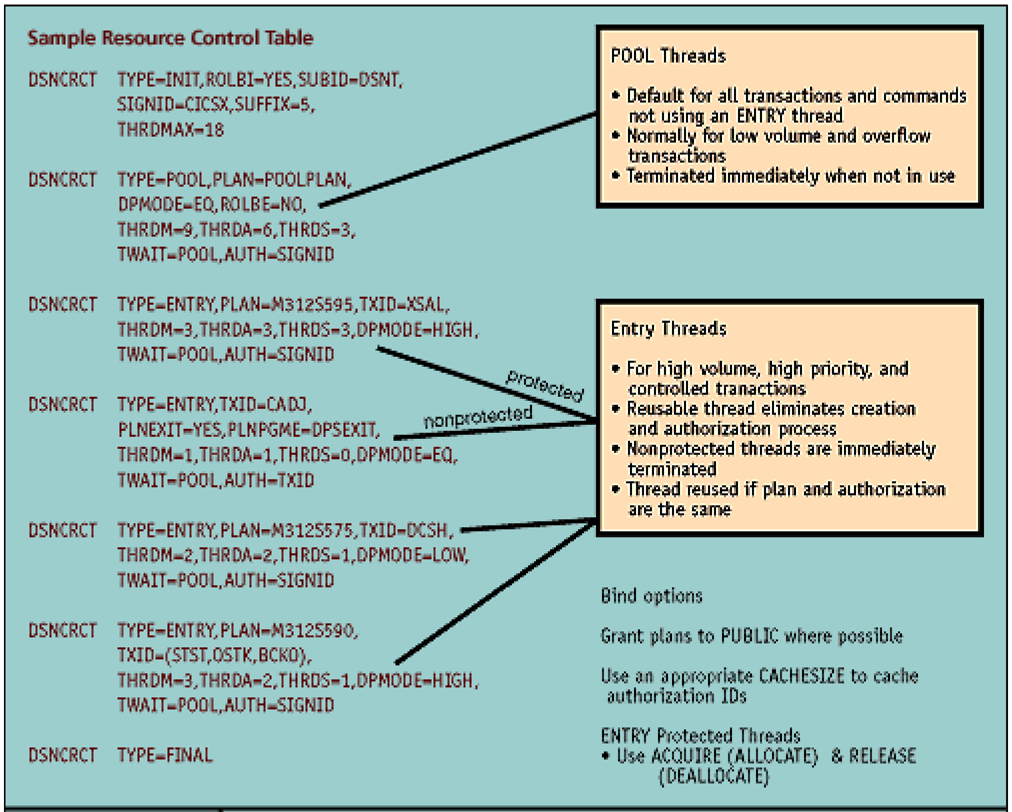
The first things you need to understand when tuning the RCT are the four phases of DB2 CICS transaction processing: sign-on processing, SQL statement processing, commit processing, and thread termination (see Table 1). Each of these activities has a different processing cost and, based on the RCT parameter settings, will process differently. For example, most DB2 CICS workloads do not need to verify user security profiles for every transaction. In most shops, transaction security is handled through RACF or ACF2 security packages.

Table 1. The four Phases of CICS transaction processing.
Signon Processing • Authorization checking • Plan security • AUTH setting checking **SQL statement processing** • Bind plan (if necessary) • Load SKCT & DBDs into EDM pool • Lock tablespace (if necessary) • Open datasets (if necessary) **Commit Processing** • Write log records • Release locks • Free cursor

Thread termination

- Write log records
- Release locks
- Free cursor
- Free working storage
- Close datasets (if necessary)

To avoid the extra overhead of rechecking security, set the RCT AUTH parameter to the appropriate transaction ID, security package group ID, or user ID. To avoid double or triple checking your security, choose the setting that makes your DB2 CICS configuration check authorization just enough for your security department. To avoid security checking, have your DB2 Plan GRANTed to PUBLIC, and make your RCT AUTH setting equal to the SIGNID in the CICS INIT section. In Figure 1, this configuration is used for the third DSNCRT transaction definition for TXID=XSAL.



Sample Resource Control Table

```
DSNCRCT   TYPE=INIT,ROLBI=YES,SUBID=DSNT,
          SIGNID=CICSX,SUFFIX=5,
          THRDMAX=18

DSNCRCT   TYPE=POOL,PLAN=POOLPLAN,
          DPMODE=EQ,ROLBE=NO,
          THRDM=9,THRDA=6,THRDS=3,
          TWAIT=POOL,AUTH=SIGNID

DSNCRCT   TYPE=ENTRY,PLAN=M312S595,TXID=XSAL,
          THRDM=3,THRDA=3,THRDS=3,DPMODE=HIGH,
          TWAIT=POOL,AUTH=SIGNID

DSNCRCT   TYPE=ENTRY,TXID=CADJ,
          PLNEXIT=YES,PLNPGME=DPSEXIT,
          THRDM=1,THRDA=1,THRDS=0,DPMODE=EQ,
          TWAIT=POOL,AUTH=TXID

DSNCRCT   TYPE=ENTRY,PLAN=M312S575,TXID=DCSH,
          THRDM=2,THRDA=2,THRDS=1,DPMODE=LOW,
          TWAIT=POOL,AUTH=SIGNID

DSNCRCT   TYPE=ENTRY,PLAN=M312S590,
          TXID=(STST,OSTK,BCKO),
          THRDM=3,THRDA=2,THRDS=1,DPMODE=HIGH,
          TWAIT=POOL,AUTH=SIGNID

DSNCRCT   TYPE=FINAL
```

POOL Threads

- Default for all transactions and commands not using an ENTRY thread
- Normally for low volume and overflow transactions
- Terminated immediately when not in use

Entry Threads

- For high volume, high priority, and controlled tranactions
- Reusable thread eliminates creation and authorization process
- Nonprotected threads are immediately terminated
- Thread reused if plan and authorization are the same

Bind options

Grant plans to PUBLIC where possible

Use an appropriate CACHESIZE to cache authorization IDs

ENTRY Protected Threads
- Use ACQUIRE (ALLOCATE) & RELEASE (DEALLOCATE)

FIGURE 1. *Sample Resource Control Table*

Figure 1 shows a small RCT with all the various types of thread definitions. These thread definitions are controlled by the following parameters:

- **AUTH:** specifies the authorization associated with the transaction connection
- **DPMODE:** specifies HIGH, EQ, or LOW level of CICS TCB execution priority
- **THRDA:** specifies the number of active connections allowed for a transaction

- **THRDM:** specifies the maximum number of active connections allowed for a transaction
- **THRDS:** specifies the number of connections to be allocated at RCT startup time for this transaction
- **TWAIT:** specifies the course of action when all connections are in use
- **TXID:** specifies the CICS transaction ID associated with the thread definition.

The first DSNCRCT block of TYPE=INIT specifies the framework for the entire RCT definition. The INIT control block has several unique parameters that are not used with the transaction thread definitions within the RCT. The first parameter (ROLBI=YES) dictates the default of the rollback activity for deadlocks or time-outs. The DB2 subsystem is specified as DSNT in the SUBID=DSNT setting. The thread authorization ID or correlation ID of CICSX is specified by SIGNID=CICSX; the RCT name or suffix character, say 5, is specified by SUFFIX=5; and the maximum number of threads, say 18, that can be active through this RCT definition is specified by the THRDMAX=18 parameter. The maximum number of threads in the RCT is very important. Without enough threads defined, the CICS system cannot pass SQL work requests to DB2. A lack of threads will have a huge negative impact on performance.

The next DSNCRCT definition block of TYPE=POOL specifies the DB2 Plan name (PLAN=M312S595), the dispatching priority (DPMODE=EQ), the rollback option (ROLBE=YES), and the maximum number of pool threads to allow (THRDM=9), allocate (THRDA=6), and start (THRDS=3). The definition for the pool must also have the TWAIT=YES parameter, because there is nowhere else for pool transactions to be serviced. The AUTH parameter specifying SIGNID refers back to the INIT block where the thread authorization ID or correlation ID of CICSX was specified. This reference to the SIGNID allows the transaction threads to be associated with the CICSX ID, possibly avoiding rechecking security and reusing the same thread. Because all transactions do not have an ENTRY definition default to the pool, it's vital that you have enough pool threads to make sure your transactions are not waiting for resources.

The main goal in tuning the DB2 CICS RCT is to minimize required activities such as CICS and DB2 security-checking and thread-creation costs. Three components determine the DB2 CICS security processing: the AUTH RCT parameter setting, the DB2 Plan, and its CACHE and GRANT execution options.

The AUTH parameter can have a variety of settings. The objective is to keep the authorization ID as constant as possible for a given transaction ID to avoid authorization checking that requires accessing the DB2 catalog. The AUTH setting possibilities that stay constant are the transaction ID (TXID), a character string, or the CICS system authorization ID (SIGNID). The other AUTH settings available -- user logon ID (USERID), sign-on operator ID (USER), or terminal ID (TERM) -- are likely to change from transaction to transaction and could cause DB2 catalog authorization. By keeping the AUTH value constant, you can avoid accessing the DB2 catalog for each transaction. If the authorization ID changes, the DB2 CICS thread could be deallocated and a new one created through thread authorization.

The DB2 Plan CACHE and GRANT execution options are other ways to avoid the overhead costs of checking security. When binding a DB2 Plan, your cache size parameter setting determines the amount of memory allocated to store user IDs approved to execute the DB2 Plan. This cache allows DB2 to avoid rechecking authorization in the DB2 catalog. Authorizing a DB2 Plan to everyone by granting execution authority to PUBLIC is another way to avoid rechecking the DB2 catalog for DB2 Plan execution authority. Although these CACHE and GRANT techniques can avoid authorization checking within DB2, the DB2 CICS thread may still be deallocated and a new one created by encountering a new or different authorization transaction ID.

The next RCT DSNCRCT definition blocks are TYPE=ENTRY threads. There are two types of ENTRY threads: protected and nonprotected. Both types can be reused so that you encounter the cost of thread creation and authorization only once for tens or hundreds of thousands of transaction executions. Because every time a thread is created it must go through authorization, minimizing thread creation reduces CPU costs and improves response time performance, making ENTRY thread definitions the best for your most active transactions.

The difference between protected and nonprotected threads is that protected threads have both the number of threads started/identified (THRDS) and allocated (THRDA) greater than zero, while nonprotected threads have only the number allocated (THRDA) greater than zero. This difference in definition allows protected threads to be started and allocated while nonprotected threads are not started. When protected and nonprotected transaction threads have completed their processing and remained inactive for approximately 45 seconds, the nonprotected threads will be terminated. Protected ENTRY thread definitions allow the protected allocated threads to be constantly ready and available for DB2 CICS workloads. The protected threads are then avoiding thread creation costs and -- depending on the AUTH settings --possibly security authorization costs.

To minimize reallocating the DB2 Plan and database descriptor (DBD) information during thread creation, bind your ENTRY thread Plans with ACQUIRE(ALLOCATE) & RELEASE (DEALLOCATE). These settings allow the thread to retain the Plan and DBD information rather than reacquire it when another SQL statement is executed. This saves CPU time and lock communications time everywhere, especially in DB2 data sharing configurations.

Using these techniques to avoid security authorization and activate thread reuse in DB2 CICS workloads can have a dramatic effect on CPU and response time. These simple but often overlooked techniques shaved .09 CPU seconds and 1/8 of a second response time off my major DB2 CICS transactions across several systems. Such tiny savings add up to big CPU and response time savings when the tuned transactions are executed millions of times per day.

Make sure you take DB2 CICS CPU and response time statistics information before your tuning effort starts. Advertise that you're working very hard on saving CPU and response time. Also, do it yourself: This type of tuning doesn't need a consultant. Besides, by doing it yourself, you can ask for a raise after your tuning is complete. Please let me know how much CPU and response time you save.

David Beulke is currently the director of corporate data management at Bell Atlantic Inc. You

can reach him at dbeulke@compuserve.com .