Natural CICS Performance Considerations

This chapter contains guidelines for setting up Natural in a CICS environment.

It covers the following topics:

- Enironment-Specific Considerations
- Choosing the Roll Facility
- Shared Storage Threads versus GETMAINed Threads
- CICS Parameter Settings
- Line Compression Systems
- Pseudo-Conversational versus Conversational Transactions
- Natural and Adabas
- CICS Monitoring Products

Enironment-Specific Considerations

The following environment-specific considerations should be noted:

- When running Natural in a CICSplex environment (z/OS only), you must use the Natural Roll Server.
- When running Natural locally in a single CICS region, however, you have several possibilities.

One possibility (z/OS only) is to use the Natural Roll Server. The benefit of this versus using CICS roll facilities and a swap pool is that the Natural Roll Server runs asynchronously to the CICS region and can provide more roll buffers in its data space than the swap pool.

Choosing the Roll Facility

This section covers the following sections:

- Control Interval
- VSAM Roll Files versus CICS Temporary Storage
- Using CICS Auxiliary Temporary Storage
- Using CICS Main Temporary Storage
- Using VSAM RRDS Roll Files

- Using the Natural Swap Pool under CICS

Control Interval

You are strongly recommended to define both roll facilities, VSAM and auxiliary temporary storage, with the largest possible control interval size of 32 KB. This minimizes the number of I/Os and the CPU overhead necessary to perform the rolling.

Reasons for a control interval size of less than 32 KB might be the better exploitation of disk tracks or the usage of virtual storage for the VSAM buffers.

VSAM Roll Files versus CICS Temporary Storage

With the same CISIZE/record size, temporary storage causes less CPU overhead than VSAM roll files:

To write *n* records to temporary storage you have to issue *n* +1 CICS requests (that is, 1 for `DELETQ` and *n* for `PUTQ`) while you have to issue 2 *n* requests for VSAM roll files because of the VSAM transaction logic: *n* times (`READ` for `UPDATE` plus `REWRITE`).

For VSAM update requests, a physical I/O is always performed, whereas for temporary storage (`AUX`) records, buffering takes place, so that in many cases, records to be read are still found in the buffers.

However, CICS temporary storage may become a bottleneck when it is also being used by other applications.

VSAM roll files for Natural can overcome this situation (although at the expense of additional VSAM buffer space) especially when I/O contention can be avoided. VSAM roll files with optimum/maximum CISIZE/record size are particularly efficient when this record size cannot be specified for the CICS temporary storage file due to other requirements.

CICS temporary storage should be used whenever it can be dedicated to Natural. If CICS temporary storage is also used by other applications, you should evaluate whether the Natural performance is better when using VSAM roll files.

If Natural with CICS temporary storage does not perform worse, you should choose CICS temporary storage as roll facility and use the "saved" VSAM roll file buffer space for more TS buffers or for an additional thread.

Using CICS Auxiliary Temporary Storage

The primary roll facility is VSAM RRDS; the default type of temporary storage is `AUXILIARY`.

If you are using VSAM roll files, the Natural CICS interface uses temporary storage (`AUX`) if all roll files become full or unusable during a CICS session.

However, if you do not wish to use roll files or if the roll files are incorrectly installed, Natural under CICS uses temporary storage (`AUX`) for all rolling. When temporary storage (AUX) is used as roll file, the control interval size for this file must be at least 4 KB. If auxiliary temporary storage is not available, main temporary storage is used instead.

The number of VSAM buffers defined by the CICS SIT parameter TS should be increased to a reasonable value to reduce the number of physical I/Os. The CICS statistics should be checked for bottlenecks in this area.

Using CICS Main Temporary Storage

With CICS main temporary storage as roll facility, no I/O is performed on rolling, but due to large main storage amounts used, tuning considerations may be required due to increased paging.

Using VSAM RRDS Roll Files

The VSAM roll files should be considered for normal CICS VSAM file tuning, for example, BUFNO and STRNO parameters in the FCT. The CICS shutdown statistics should be checked for bottlenecks in this area.

As the roll files serve as a kind of page dataset for Natural, everything which slows down the Natural rolling should be avoided, as there is journaling and logging; dynamic transaction backout (DTB) and forward recovery for roll files is useless and only causes overhead.

In MRO Environments

For performance reasons the VSAM roll files should be defined in the same CICS system in which the Natural applications are running; MRO function shipping should not be invoked. CICS local shared resources (LSR) can be used if there are enough buffers available.

Separate LSR Pool for Natural

The definition of a separate LSR pool for Natural roll files is recommended, with the number of strings (STRNO) greater than the number of threads. The number of buffers should also be greater than the number of threads. A greater number of buffers increase the look-aside hit ratio.

Using the Natural Swap Pool under CICS

You are strongly recommended to use a swap pool rather than a large number of VSAM temporary storage (AUX) buffers or temporary storage (MAIN).

The Natural swap manager handles the compressed session storage very efficiently and reduces CPU and I/O overhead. The size of the swap pool should be as large as possible. For example, a swap pool of 2.5 MB would be required to hold 50 sessions which fit into 50 KB slots.

From a performance point of view, it does not make any sense to use main temporary storage as a backup facility for the swap pool, since both of these facilities use CICS main storage. In general though, using the swap pool is more advantageous, because CICS services overhead is eliminated. Rather than overflowing to main temporary storage, it would be better to enlarge the swap pool and to use disk storage (that is, VSAM roll files or auxiliary storage) as its backup facility.

If virtual storage becomes a bottleneck, the number of roll facility buffers and possibly the number of threads should be minimized to the benefit of the swap pool.

When using the Natural swap pool cache, a roll buffer of the size of the largest Natural thread is required for transferring Natural session data between the swap pool and its (data space) cache. This roll buffer is taken from the GETMAIN for the swap pool, that is, the size of the storage actually available for the swap

pool is the specified size minus the size of the largest Natural thread.

Therefore a Natural swap pool cache is only allocated when both the size of the swap pool and the size of its cache are at least twice the size of the largest Natural thread.

Shared Storage Threads versus GETMAINed Threads

This covers section the following sections

- Storage Usage
- Controlling Storage Usage
- Swapping/Rolling
- Considerations for CICS/TS
- Conclusion

Storage Usage

Shared storage threads are pre-allocated during Natural CICS system initialization, which means that the storage covered by these threads is dedicated to the Natural CICS system, regardless of whether there are active sessions or not. On the other hand, `GETMAIN`ed threads only exist while the CICS task is active.

Controlling Storage Usage

With shared storage threads (`TYPE=SHR`), Natural under CICS always uses what has been pre-allocated during the initialization of Natural; therefore, the size of storage used by Natural threads is easily predictable. For `GETMAIN`ed threads (`TYPE=GETM`), however, the actual storage used depends on the number of Natural sessions that are currently active.

Although Natural itself has no mechanism for setting the maximum number of `GETMAIN`ed threads, this can be controlled by grouping the Natural transaction codes into a `TRANCLASS` (`TCLASS` prior to CICS Version 4.1). When a transaction code belongs to such a class, the maximum number of parallel tasks can be regulated by the `MAXACTIVE` parameter in the `TRANCLASS` definition (or by using the `CMXT` parameter of the CICS system initialization table (SIT) prior to CICS Version 4.1).

Swapping/Rolling

When a Natural session releases its shared storage thread, session data are kept in the thread in uncompressed format, unless another session needs to use this particular thread. If so, the new session is responsible for saving the old session’s data.

Such an activity is called "deferred rolling". It enables you to eliminate rolling or swapping entirely, provided that the number of available threads is greater or equal to the number of concurrently active Natural sessions.

Conversely, sessions that use `GETMAIN`ed threads always save their data prior to the `FREEMAIN` operation at CICS task termination, which leads to a roll/swap overhead regardless of the number of concurrently active Natural sessions.

In environments with high volumes of Natural transactions, there is practically no difference between saving session data via the "immediate" or the "deferred" rolling method.

In busy Natural environments with a high ratio of Natural sessions to program storage threads, there is more roll-in/roll-out overhead, since these threads are shared by several Natural sessions. A potential problem in this situation is thread contention caused by Natural tasks with long-running Adabas requests; that is, with many Adabas calls.

To prevent such tasks from "locking" a thread for too long, they can be forced to release their thread by using Natural profile parameter `DBROLL` appropriately.

For `GETMAIN`ed threads, however, contention between two or more Natural sessions never occurs, since a `TYPE=GETM` thread belongs exclusively to the Natural session it was allocated for.

`TYPE=GETM` threads can thus be considered "single-use" resources that are never shared, whereas `TYPE=SHR` threads can be considered "multi-use" resources that may be shared.

Considerations for CICS/TS

The most important feature of CICS/TS in z/OS is transaction isolation, which means that a task's storage can be protected against other tasks.

To take advantage of this facility with Natural, `TYPE=GETM` threads should be used, since these threads are subject to transaction isolation, whereas "shared" `TYPE=SHR` threads are not. Also additional CICS overhead occurs for `TYPE=SHR` threads with CICS/TS.

While the thread selection algorithm for `TYPE=GETM` threads is trivial (when a Natural task is started, a thread is allocated via CICS `GETMAIN`), for `TYPE=SHR` threads, it is more complicated: the Natural threads environment is managed by `NCISTART` (queueing and balancing), whereas CICS does not know anything about Natural threads. In contrast to `TYPE=GETM` threads, where CICS would release the thread at the latest at the end of the task, with `TYPE=SHR` threads, Natural has to assign/release them to/from their sessions. In order to do so, Natural maintains a list of thread control blocks (TCBs).

Although Natural always keeps an exit active to be able to release session resources unknown to CICS (for example, `TYPE=SHR` threads) in the case of abnormal task termination, situations may occur where a Natural task terminates without its thread being marked as free in the associated TCB (for example, if an `EXEC CICS ABEND CANCEL` request has been issued in a non-Natural program called by Natural, or if Natural sessions have been flushed by any `KILL` transactions of a performance monitor).

To prevent problems with threads inadvertently left busy, Natural under CICS always checks in its thread selection algorithm whether the CICS task associated to a busy thread is still existing; if not, the thread is released.

With CICS versions prior to CICS/TS, this checking for active CICS tasks was done by control-block jumping, which means that Natural was checking for an active task by testing the consistency of the task's EISTG, TCA and TQE control blocks. With CICS/TS, because of transaction isolation, the storage of another task may not be accessible at all.

To accomplish this function in CICS/TS, NCISTART issues an `EXEC CICS INQUIRE STORAGE TASK()` request for any thread identified as busy in the thread selection routine. This means that there may have been some CICS requests before the task is finally ENQueued for thread resources. The same CICS command is also used for the serialization of Natural sessions (for example, deferred rolling of `TYPE=SHR` threads).

Conclusion

Both TYPE=SHR and TYPE=GETM threads have their advantages and disadvantages. However, with CICS/TS, TYPE=GETM threads are preferred, because of:

- the support of transaction isolation (z/OS only),
- more CICS-like tuning possibilities,
- worse performance of TYPE=SHR threads.

CICS Parameter Settings

CICS SIT parameters AMXT and CMXT should be used to control the number of concurrent Natural tasks.

The number specified should be greater than the number of threads. You should also consider to specify a separate transaction class with a suitable CMXT parameter for asynchronous Natural tasks and for Natural Advanced Facilities spool tasks so as to prevent logouts of "normal" Natural terminal tasks by too many of such "background" tasks occupying threads. Special thread groups can be defined for these transactions.

CICS dumps for Natural transactions should be suppressed, unless requested from Software AG personnel for debugging purposes. Natural itself generates dumps (via EXEC CICS DUMP) for non-program check abends, and also for program checks if the Natural session parameter DU is set to ON. When no Natural dump is generated, a CICS dump is redundant and just causes overhead (particularly when creating a system/region dump, since the whole CICS system is halted until the snap dump is completed).

CICS trace is essential when analyzing problems, although it introduces system overhead. Also CICS performance monitoring tools and accounting packages cause system overhead of up to 30 percent and more. Some packages internally turn on the CICS trace and then intercept it. You should be aware of this potential system overhead. Also remember that the Natural CICS interface uses the CICS command level application programming interface: CICS command level requests produce much more trace entries (apart from other overhead) than CICS macro level requests.

Line Compression Systems

Natural itself optimizes its data streams by means of RA (repeat to address) and other techniques as screen imaging etc. If other line compression systems are installed, the Natural transactions should be excluded from being processed by these systems, as this would introduce overhead without achieving any benefit.

Pseudo-Conversational versus Conversational Transactions

When resuming a session, conversational Natural tasks are locked to their initial thread, which means that a conversational task has to wait for this thread if it is currently not available. Pseudo-conversational Natural tasks, however, are flexible to roll into any available thread.

In other words, the "classical" advantage of conversational tasks - less I/Os for saving/restoring data over screen I/O operations - does not apply for Natural because of its thread technique.

Natural and Adabas

Since a Natural task in CICS waits for completion of an Adabas call, the servicing Adabas region/partition should always have higher priority than the CICS region/partition to minimize wait time.

CICS Monitoring Products

CICS monitoring products may offer facilities to purge CICS tasks, bypassing any abnormal termination exit set by the application.

Warning:
Such facilities should not be used to cancel Natural tasks, as Natural may not be able to clean up its resources, and, even worse, the Natural CICS system may be left in an inconsistent state depending on what this task was doing.

To cancel Natural sessions, the Cancel/Flush Session functions of the Natural SYSTP utility should be used instead; see the relevant section in the Natural *Utilities* documentation for details.