
Rule WLM123: Significant transaction time was in Waiting for Lock state

Finding: A significant amount of the transaction response time for the service class missing its performance goal was spent in the Waiting for Lock state. This finding applies to service classes which are part of a subsystem (e.g., CICS transactions).

Impact: This finding has MEDIUM IMPACT or HIGH IMPACT on performance of the service class. The level of impact depends on the percent of transaction response time spent in the Waiting for Lock state.

Logic flow: The following rules cause this rule to be invoked:

- Rule WLM104: Subsystem Service Class did not achieve average response goal
- Rule WLM105: Subsystem Service Class did not achieve percentile response goal

Discussion: When CPExpert produces Rule WLM104 or Rule WLM105 to indicate that a subsystem service class did not achieve its performance goal, the logic of these rules tries to identify the cause of the delay. The cause of the delay initially is analyzed from the "served" service class view. The delays from the served service class are reported by CICS/ESA Version 4.1 or IMS Version 5 interaction with the Workload Manager, using the Workload Management Services macros¹.

CICS/ESA Version 4.1 reports two separate views of the transactions: the *begin_to_end phase* and the *execution phase*².

- **Begin_to_end phase.** The *begin_to_end phase* starts when CICS/ESA Version 4.1 has classified the transaction³. This action normally is done in a CICS Terminal Owning Region (TOR).
- **Execution phase.** The *execution phase* starts when either CICS/ESA Version 4.1 or IMS Version 5 has started an application task to process

Please refer to Section 4 of this document for more detail about the Workload Management Services macros and how the subsystems use these macros to exchange information with the Workload Manager.

IMS Version 5 reports only *execution phase* samples.

Classifying the transaction into a service class is actually done by the Workload Manager when CICS issues the IWMCLSFY macro. Please refer to Section 4 for a more complete discussion of the subsystem work manager (e.g., CICS) interaction with the Workload Manager.

the transaction. For CICS, this normally is done in a CICS Application Owning Region (AOR).

Within each phase, CICS or IMS reports the "state" of the transaction, from the view of CICS or IMS. The state of the transaction is reported in the following categories⁴:

- **Idle state.**
- **Active state.**
- **Ready state.**
- **Wait state.**
- **Switched state.**

If the subsystem supports work manager delay reporting, the delay information is available in the "Work Manager/Resource Manger State Section" of SMF Type 72 (Subtype 3) records. When a transaction service class fails to achieve its performance goal, CPExpert analyzes the information to identify the primary and secondary causes of delay.

The Wait state indicates that a task in support of the transaction was waiting on some activity. The Wait state is broken into several categories: waiting for lock, waiting for I/O, waiting for conversation, waiting for distributed request, waiting for a session to be established (locally, somewhere in the network, or somewhere in the sysplex), waiting for a timer, waiting for another product, or waiting for an unidentified resource.

CPExpert produces Rule WLM123 when the primary or secondary cause of delay was that the transaction service class was in the Waiting for Lock state for a significant percent of its response time.

For CICS transactions, this is the time accounted for by tasks which were suspended waiting for such locks as:

- A lock on a CICS resource.
- A record lock on a recoverable VSAM file.
- Exclusive control of a record in a BDAM file

Please refer to Section 4 of this document for a more comprehensive discussion of the transaction states and the interaction between the subsystem (CICS or IMS) and the Workload Manager.

-
- An application resource that has been locked by an EXEC CICS ENQ command

These tasks would be shown as "Suspended" by the CEMT INQUIRE TASK command.

The following example illustrates the output from Rule WLM123:

```
RULE WLM123:  SIGNIFICANT TRANSACTION TIME WAS WAITING FOR LOCK

A significant amount of the transaction response time for CICUSRB Service
Class was spent in the Waiting for Lock State.  For CICS transactions,
this is the time accounted for by tasks which were suspended waiting
for such locks as:
- A lock on a CICS resource.
- A record lock on a recoverable VSAM file.
- Exclusive control of a record in a BDAM file
- An application resource that has been locked by an EXEC CICS
  ENQ command
These tasks would be shown as "Suspended" by the CEMT INQUIRE TASK
command.
```

Suggestion: IBM has provided detailed information about the Workload Manager I/O Wait types used by CICS. Exhibit WLM123-1 shows the resources that a suspended task might be waiting on for the Workload Manager Lock Wait type.

Many of the causes of time spent Waiting for Lock are related to application design, and the solutions often require a review of the approach to the application or file design.

As shown in Exhibit WLM123-1, there are seven reasons that CICS provides the Workload Manager with a Wait for Lock.

TYPE OF WAIT	TYPE OF TASK	RESOURCE TYPE	RESOURCE NAME	SUSPENDING MODULE
CICS system task waits	System task	AP_TERM	STP_DONE	DFHAPDM
File control waits	User task	KC_ENQ	SUSPEND	DFHXCPD
Loader waits	User task	PROGRAM	program_ID	DFHLDL
Lock manager waits	User task	(none)	LMQUEUE	DFHMLM
Task control waits	User task	KCCOMPAT	CICS	DFHXCPC
Task control waits	User task	KC_ENQ	SUSPEND	DFHXCPD
Temporary storage wait	User task	TSBUFFER	(none)	DFHTSP
Temporary storage wait	User task	TSEXTEND	(none)	DFHTSP
Temporary storage wait	User task	TSOPEN4B	(none)	DFHTSP
Temporary storage wait	User task	TSQUEUE	(none)	DFHTSP
Temporary storage wait	User task	TSSTRING	(none)	DFHTSP
Temporary storage wait	User task	TSUT	(none)	DFHTSP
Temporary storage wait	User task	TSWBUFRR	(none)	DFHTSP
Transient data waits	User task	KC_ENQ	SUSPEND	DFHXCPD
Transient data waits	User task	TDEPLOY	transient	DFHTDEXP
Transient data waits	User task	TDILOCK	transient	DFHTDSUB

CICS WAITING FOR LOCK Exhibit WLM123-1

- CICS system task waits.** CICS module DFHAPDM is the Application Domain (AP) module responsible for initializing, quiescing, and terminating the application domain. CICS provides the Workload Manager with a Wait for Lock when the application domain is being terminated (shutdown or takeover). This lock type would not cause an individual transaction to miss its performance goal.
- File Control waits.** Lock waits caused by file control can occur when a task is waiting for a record lock in a recoverable VSAM file. When an application updates a record in a recoverable VSAM file, locking occurs at two levels: (1) VSAM locks the Control Interval (CI) when the record has been read, and (2) CICS locks the record.

The CI lock is released as soon as the REWRITE (or UNLOCK) request is completed. However, the record is not unlocked by CICS until the updating task has reached a syncpoint. This is to ensure that data integrity is maintained if the task fails before the syncpoint and the record has to be backed out.

If a second task attempts to update the same record while the record is still locked, the second task is suspended on resource type KC_ENQ until the lock is released. This can be a long wait, because the update might depend on a terminal operator typing in data. Also, the suspended task relinquishes its VSAM string and may relinquish its exclusive control

of the CI. The suspended task would have to regain these resources and may have to wait after it was no longer in a file control lock wait.

BDAM does not use the "control interval" concept. When a task reads a record for update, the record is locked so that concurrent changes cannot be made by two transactions. The lock is released at the end of the current logical unit of work. If a second task attempts to update the same record while the first has the lock, it is suspended on resource type KC_ENQ.

Solving Lock Wait due to file control may require a review of the application logic or file design to see if the record-locking time can be reduced.

- **Loader waits.** A task is suspended by the loader domain if it has requested a program load and another task is already loading that program. Once the load in progress is complete, the suspended task normally is resumed quickly and the wait is unlikely to be detected.

If the requested program is not loaded quickly, there are two likely causes:

- **The system could be short on storage (SOS)**, so only system tasks can be dispatched. The Storage Manager Statistics part of the CICS interval statistics contain information which can be analyzed to determine whether the WLM Lock wait was likely caused by a SOS condition. The field SMSSOS is a count of the number of times CICS went SOS in a particular subpool (note that there are separate statistics for each of the storage subpools).
 - If the SMSSOS value is zero, you can be sure that the WLM Lock waits were **not** caused by Loader waits.
 - If the SMSSOS value is non-zero, it is possible that the WLM Lock waits **were** caused by Loader waits because CICS entered SOS. Unfortunately, there is no way to determine whether a task suspended for a Loader wait actually was in the service class missing its performance goal. However, the CICS region was encountering SOS, and you should take action.

If the SMSSOS value is non-zero, CPExpert suggests that you review the suggested actions beginning on page 251 of the IBM *CICS Version 4.1 Performance Guide*. These actions provide a checklist for reducing the virtual storage requirements above and below the 16MD line.

Alternatively, you can execute the CICS Component of CPExpert against the CICS region(s) serving the service class missing its performance goal. The CICS Component will analyze the CICS interval statistics to identify performance problems.

- **There could be an I/O error on a library.** You can check for messages that might indicate an I/O error on a library. If you find that an I/O error occurred, you should investigate the reason why the I/O error occurred.
- **Lock Manager waits.** The Lock Manager suspends a task when the task cannot acquire the lock on a resource it has requested, probably because another task has not released it. A user task cannot explicitly acquire a lock on a resource, but many of the CICS modules that run on behalf of user tasks do lock resources. Lock Manager waits could indicate a CICS system error.

You should review the "Lock Manager Waits" part of Section 2.3: Dealing with waits (Bookmanager document) of the CICS/ESA Version 4.1 Problem Determination Guide.

While it is possible to experience Lock Manager waits, it is unlikely that these are the cause of performance problems with the service class missing its performance goal.

- **Task Control waits.** Task Control will suspend a task (1) if the task has attempted to change the state of a file but another task is still using the file, (2) if the task attempted to update a record in a recoverable file while another task has a lock on the file, or (3) if a task has finished using a file but not issued an EXEC CICS DEQ command or a DFHKC TYPE=DEQ macro call.

Solving these problems require a review of the approach to the application or file design.

- **Temporary storage waits.**
 - Resource type TSBUFFER indicates that the task that is waiting has issued an auxiliary temporary storage request, but the buffers are all in use. If you find that tasks are often made to wait on this resource, consider increasing the number of auxiliary temporary storage buffers (system initialization parameter TS).
 - Resource type TSEXTEND indicates that the waiting task has issued a request to extend the auxiliary temporary storage data set, but some other task has already made the same request. The wait does

not extend beyond the time taken for the extend operation to complete. If you have a task that is waiting for a long time on this resource, it is likely that there is a hardware fault or a problem with VSAM.

- Resource type TSQUEUE indicates that the waiting task has issued a request against a temporary storage queue that is already in use by another task. The latter task is said to have the lock on the queue.

The length of time that a task has the lock on a temporary storage queue depends on whether or not the queue is recoverable. If the queue is recoverable, the task has the lock until the logical unit of work is complete. If it is not recoverable, the task has the lock for the duration of the temporary storage request only.

- Resource type TSSTRING indicates that the task is waiting for an auxiliary temporary storage VSAM string. If you find that tasks frequently wait on this resource, consider increasing the number of temporary storage strings (system initialization parameter TS).
- If a user task is waiting on resource type TSUT, activity keypointing is taking place. This involves a large amount of I/O, and, if there are many temporary storage queues, it could take a relatively long time to complete.
- Resource type TSWBUFFR indicates that the waiting task has issued an auxiliary temporary storage request, but the write buffers are all in use. You have no control over how temporary storage allocates read buffers and write buffers from the buffer pool, but if you find that tasks are often made to wait on this resource, increasing the number of auxiliary temporary storage buffers (system initialization parameter TS) should help solve the problem.
- **Transient data waits.** Transient data waits occur when a task is suspended on resource type TDEPLOCK, with a resource name corresponding to a transient data queue name. The task has issued a request against an extrapartition transient data queue, but another task is already accessing the same queue. The waiting task cannot resume until that activity is complete.

Significant time spent in transient data waits occur because it is necessary for a task to change TCB mode to open and close a data set. The task must relinquish control while this happens. Depending on the system loading, relinquishing control might take several seconds. This contributes to the wait that the second task experiences, while the second task is suspended on resource type TDEPLOCK,.

CICS uses the access method QSAM to write data to extrapartition transient data destinations. QSAM executes synchronously with tasks requesting its services. This means that any task invoking a QSAM service must wait until the QSAM processing is complete. If, for any reason, QSAM enters an extended wait, the requesting task also experiences an extended wait.

The possibility of an extended wait arises whenever QSAM attempts to access an extrapartition data set. QSAM uses the MVS RESERVE volume-locking mechanism to gain exclusive control of volumes while it accesses them, which means that any other region attempting to write to the same volume is forced to wait.

If tasks frequently get suspended on resource type TDEPLOCK, you should determine which other transactions write data to the same extrapartition destination. You might then consider redefining the extrapartition destinations in the DCT (destination control table).

You can find further guidance information about the constraints that apply to tasks writing to intrapartition destinations in the CICS Application Programming Guide. For more details of the properties of recoverable transient data queues, see the CICS Resource Definition Guide.

- Another common cause of locks on a CICS resource is the CICS shared database facility. An IMS batch job can access a local DL/I database controlled in a CICS region. Any DL/I request from the IMS batch application program is handled through the facilities of CICS instead of IMS DB.

A shared database region contains an IMS batch application program that processes local DL/I databases, and the application program in the shared database region is scheduled by MVS job management. The job stream for the job specifies the CICS batch region controller. The shared database program uses DL/I CALLS for database references. An application program executing in a shared database region can access only the local DL/I databases that are attached to the CICS online region.

The CICS shared database facility can greatly increase contention for a database, particularly if update operations from batch programs are involved.

- A normal CICS task accesses and enqueues on a small number of records from a database.

-
- An IMS batch program may access and enqueue on all the records in the database, effectively locking up the database until the program completes⁵. If the batch jobs are update jobs, they are likely to lock out the database from online use until they finish running, which typically takes several minutes.

The following guidance is provided by IBM in the referenced *CICS Performance Guides*:

- CICS using DBCTL performs better than function shipping. Performance can be improved by replacing any database owning region (DOR) with a DBCTL owning region.
- Users accessing DL/I databases from CICS via the IMS DBCTL facility should use IMS BMPs rather than CICS shared database.
- In general, use CICS shared database only when absolutely necessary. Either try to minimize or eliminate update operations and run batch jobs during offpeak times when the system is not busy, or use IMS data sharing.

If it is necessary to run batch update during online operations, do one of the following:

- Run the batch update during periods of low online activity.
- Close down the online transactions that reference the database
- Inform users of the database that they are most likely to experience an increase in response time during the period of updating from the batch region
- Incorporate frequent checkpoints in batch applications.

You should also review all DL/I PSBs to minimize the contention between batch and online CICS transactions and possibly increase the priority for online transactions versus the partition control task.

If batch update operations are required, use of the IMS/ESA or DL/I Checkpoint Call can free up records when they are updated, but may complicate program restart in the case of a batch program abend.

This can also greatly increase the requirements for storage in the IMS/ESA enqueue pool

Storage for the dynamic buffer may need to be increased because a large amount of backout information may have to be kept until batch program completion.

Reference: CICS/ESA Version 4.1 Performance Guide
Section 2.7.1.1: The response time breakdown in percentage section
Section 2.7.1.2: The state section

CICS/TS Release 1.1 Performance Guide
Section 2.7.1.1: The response time breakdown in percentage section
Section 2.7.1.2: The state section

CICS/TS Release 1.2 Performance Guide
Section 2.7.1.1: The response time breakdown in percentage section
Section 2.7.1.2: The state section

CICS/TS Release 1.3 Performance Guide
Section 2.6.1.1: The response time breakdown in percentage section
Section 2.6.1.2: The state section

CICS/TS for z/OS Release 2.1 *Performance Guide*: Chapter 8 (Managing Workloads).

CICS/TS for z/OS Release 2.2 *Performance Guide*: Chapter 8 (Managing Workloads). |