EVALUATING AND IMPROVING CICS PERFORMANCE IN MVS (GOAL MODE)

Donald R. Deese

CICS/ESA Version 4.1 and CICS/Transaction Server for OS/390 operating under MVS (Goal Mode) can provide significant performance improvements over MVS (Compatibility Mode). In Goal Mode, these systems provide a wealth of performance information that can be analyzed to remove constraints to improved CICS performance. This paper describes the information available in Goal Mode and provides suggestions about how to improve CICS performance based on the information provided.

1.0 INTRODUCTION

Beginning with CICS/ESA Version 4.1, installations operating MVS (Goal Mode) can define service classes that describe particular CICS transaction types. The installations then can specify **response goals** for the CICS transactions executing in the service class[1]. The Workload Manager (WLM) monitors the performance of the service classes to which the CICS transactions are assigned. The WLM can manage the allocation of system resources to ***address spaces*** (CICS regions, IMS regions, and other address spaces supporting the CICS transactions) based on how well the transactions meet the response goals specified for their service classes.

In the Goal Mode lexicon, the transaction service classes are considered ***served*** service classes, and the address spaces are considered ***servers***. It is important to appreciate that the Workload Manager does not allocate resources to the **served** transaction service classes, as these service classes are not address spaces, but are simply logical entities that describe transactions. The Workload Manager allocates resources to the **server** address space (the CICS or IMS regions). Similarly, the Workload Manager does not measure resources consumed by the transaction service classes, as this information is not reported to the Workload Manager.

One implication of the structure of the *server* and *served* service classes is that the Workload Manager will attempt to meet the performance goals of all *served* service classes served by the *server* service class. It does this by allocating resources to the *server* service class. **These additional resources may (or may not) be used by CICS to provide service to the service class missing its goal**. Consequently, most of the performance evaluation of CICS executing in Goal Mode must focus on CICS internal controls. The WLM will allocate resources to the CICS regions, but internal CICS controls will dictate how the resources are used.

2.0 CICS/WLM INTERACTION

IBM provides a standardized interface between the Workload Manager and subsystems (such as CICS and IMS). Both CICS (with Version 4.1) and IMS (with Version 5.1) take advantage of the interface to provide detailed information about the internal processing of transactions.

Subsystems communicate with the Workload Manager using Workload Manager Services macros[2]. When a CICS region (Version 4.1 and above) is started under Goal Mode, the region requests that MVS allocate *Performance Blocks* for transactions in the region.

When a CICS transaction enters a CICS region, CICS requests that the WLM use the workload classification scheme to assign the transaction to a service class, assign Performance Blocks to the transaction, and update the Performance Blocks to indicate that the transaction has started. Similarly, CICS notifies the WLM when transactions end, so the transaction elapsed time can be recorded.

As transactions are processed, CICS (or IMS) will provide MVS with information about the "state" of the transaction by issuing the IWMMCHST ("Change State of Work Request") macro. MVS simply sets bits in the Performance Blocks to indicate the state of the transaction. Information from the Performance Blocks is recorded by RMF in SMF Type 72 records. Figure 1 illustrates the state of transactions from the view of both the WLM and CICS.

---

[1] It is not necessary that service classes be specified for CICS transactions when executing in Goal Mode. The CICS regions could be assigned to a service class (perhaps with an execution velocity goal) and transactions service classes might not be defined. However, the WLM will **not** manage the allocation of system resources based on transaction response time unless transaction service classes are defined. Managing the allocation of system resources based on how well transaction response times meet performance goals is a major advantage of Goal Mode. Consequently, this paper assumes that service classes have been defined for CICS transactions, and appropriate response goals have been specified for the transaction service classes.

[2] These macros are described in IBM's *Programming: Workload Management Services* document.

As shown by Figure 1, CICS reports two separate views of the transactions: the ***begin_to_end phase*** and the ***execution phase***[3].

- **Begin_to_end phase**. The begin_to_end (BTE) phase starts when MVS has classified the transaction. This classification action is normally requested in a CICS Terminal Owning Region (TOR).

- **Execution phase**. The execution phase starts when CICS has started an application task to process the transaction. Processing the transaction is normally done in a CICS Application Owning Region (AOR)[4].

Within both the begin_to_end phase and the execution phase, CICS provides information describing the state of the transaction: Ready state, Active state, Idle state, Wait state, and Switched state.

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 sysplex, or somewhere in the network), Waiting for a Timer, Waiting for Another Product, Waiting for a New Latch, or Waiting for an Unidentified Resource (Wait Miscellaneous).

The Switched state also is broken into categories that identify where the transaction has been switched (switched to another subsystem in the local MVS image, switched to another system in the sysplex, or switched somewhere in the network). All this information about the state of CICS transactions is recorded in SMF Type 72 records, and is reported by RMF.

The Workload Manager periodically assesses the performance of each service class, comparing the performance **achieved** by the service class against the performance goals **specified** for the service class[5]. The comparison of performance achieved is accomplished by computing a *Performance Index* for each service class.

For *response* goals, the Performance Index is computed by dividing the **actual** response by the response **goal**. If actual response is less than the goal, the Performance Index will be less than one. If the actual response is greater than the goal, the Performance Index will be greater than one.

The information necessary to compute the Performance Index is recorded in SMF Type 72 records, and is reported by RMF. This Performance Index can be examined to decide whether CICS transaction service classes meet their performance goal.

As described earlier, CICS reports transaction start and ending times to MVS, so MVS can calculate the elapsed time of each transaction. These transaction response times are mapped into 14 counters, with each counter describing the transaction response times relative to the performance goal specified for the service class. Transaction response distributions are recorded into SMF Type 72 records and reported by RMF.

3.0 ANALYZING CICS TRANSACTION DELAYS

Detailed CICS performance analysis should normally be done only for transaction service classes that miss their performance goal (that is, they have a Performance Index greater than 1.0). The response time distributions for these CICS transaction service classes can be examined to understand how serious the performance problem is, and how much time should be spent in analyzing transaction delays.

As illustrated in Figure 1, CICS dynamically reports the state of each CICS transaction to MVS, and these transaction states are available for analysis in RMF reports.

The following paragraphs describe the states for the CICS transaction service classes shown in Figure 1, and suggest ways to analyze the transaction states. **The analysis should focus only on those service classes missing their performance goals, and alternative actions should be considered only for the specific states in which a significant amount of transaction time appeared**.

3.1 READY STATE. The Ready state indicates that a program was ready to execute on behalf of a work request in the "served" service class, but that the subsystem has given priority to another work request. For a CICS region, this means that more CICS tasks were ready to process transactions in the "served" service class than were dispatched by CICS. These tasks would be shown as "Dispatchable" by the CEMT INQUIRE TASK command.

Consider the following alternatives if a significant amount of CICS transaction time is spent in the Ready State:

- **Review CICS task prioritization**. The task supporting the transaction service class that missed its performance goal was waiting for dispatch within the CICS region. One way to improve the response

[3] Note that some CICS transactions never enter the execution phase, as the transactions will be completely processed in the CICS TOR. Consequently, the number of transactions completing the execution phase may be less than the total number of CICS transactions processed by the system.

[4] The transaction might be switched to an IMS region (the IMS region would report IMS-related states, and this information would be recorded as an additional execution phase).

[5] A response goal (either an average response goal or percentile ponse goal) must be specified for CICS transaction service classes. Two other performance goals exist: execution velocity goal and discretionary goal. These goals may not be specified for CICS transaction service classes.

of important transactions is to give specific tasks preference in being dispatched by CICS.

Dispatching priority of tasks **within a CICS region**[6] is specified in three ways: (1) priority by terminal in the CEDA TERMINAL definition (the value of the TERMPRIORITY keyword), (2) priority by transaction in the CEDA TRANSACTION definition (the value of the PRIORITY keyword), and (3) priority by operator in the signon table (the value of the OPRTY keyword in the SNT). Additionally, the three priorities can be specified via the CEMT command. The overall priority is determined by summing the priorities in the three definitions for each task, with a maximum resulting priority of 255.

CICS maintains a dispatch queue of tasks that are ready to execute. The dispatch queue is ordered by priority, and CICS selects tasks from the top of the queue to dispatch.

- If task prioritization is not implemented, tasks are placed on the bottom of the queue as they become ready to execute[7]. Thus, CICS selects tasks for dispatching in the order in which the tasks become ready to execute.

- If task prioritization is implemented, a task that becomes ready to execute is placed on the queue based on its priority. A high priority task becoming ready to execute is placed on the queue ahead of all lower priority tasks, but below tasks at the same priority. Task prioritization could be used so that CICS would favor important transactions.

Task prioritization should be used sparingly, with task priority given to only the most important CICS tasks. The referenced *Performance Guides* (Task Prioritization section) explain the effects, uses, limitations, and implementation of task prioritization.

- **Remove selected transactions from the CICS region**. CICS task prioritization is not interrupt driven, as is MVS dispatching. The CICS task prioritization scheme simply relates to the relative position of tasks on the CICS dispatching queue. Once CICS has selected a task for dispatching, the task will remain dispatched until the task returns control to CICS.

  The Workload Manager allocates resources to address spaces (e.g., CICS regions), not to transaction service classes. The CICS region could be providing good service to other, less important transactions in different service classes. These service classes could be using significant system resources and delaying CICS in its dispatching of the important transactions.

  If relatively long-running tasks serve transactions with a relatively low importance, these tasks may retain control of CICS for prolonged intervals. The result might be that the transactions with a high importance are delayed waiting for CICS to select their corresponding tasks for dispatch. One further result might be that the Workload Manager would allocate more resources to the CICS region as the important transaction service classes continued to miss their goal. Unfortunately, the additional resources might not help improve performance of the important tasks since CICS internally controls dispatching of tasks and these tasks might not release control.

  A solution to this problem is to identify the transactions processed by the long-running tasks and remove them from the CICS region. This is normally the preferred approach (having a CICS region serving only the most important transactions is normally preferable). This approach might require that another CICS region be generated, however.

- **Identify "long" transactions and optimize their related tasks**. This approach could result in large benefits, but generally requires a significant amount of application programmer time.

- **Speed the flow of all transactions through the CICS region**. The CICS region operates within the standard MVS environment. The CICS region may be delayed for various reasons (CPU dispatching, I/O access, etc.). The "server" service class should be analyzed to decide whether the server (i.e., the CICS region) was using the CPU, whether the server was denied access to the CPU, etc.

3.2 ACTIVE STATE.

The Active state indicates that a task was executing on behalf of the transaction, **from the perspective of CICS**. For CICS transactions, this is the time accounted for by tasks executing in the CICS region. These tasks would be shown as "Running" by the CEMT INQUIRE TASK command. If a significant amount of CICS transaction time is spent in the Active State, analyze the "server" service class to decide whether the server (e.g., the

[6] The dispatching priority within a CICS region has no relationship to the dispatching priority from the perspective of MVS. The dispatching priority within the CICS region controls the order in which tasks are placed onto the dispatching queue in the region.

[7] Additionally, the dispatching priority can be increased based on the length of time a task has remained on the dispatching queue without being dispatched. The PRTYAGE parameter in the System Initialization Table (SIT) controls the frequency with which a task is examined to determine whether its priority should be increased. The PRTYAGE specification is in milliseconds, and directs CICS to increase the priority of a task once the task has been on the dispatch queue for the PRTYAGE duration. The default value of the PRTYAGE parameter is 32768, indicating that a task's priority will be increased by 1 when the task has been on the dispatch queue for 32,768 milliseconds.

CICS region) was using the CPU, or whether the server was denied access to the CPU.

The fact that CICS reports "Active" state does not mean that the CICS tasks are processing the transaction. MVS allocates CPU cycles based on MVS dispatching priority, and the CICS region may be denied access to a CPU. CICS might have dispatched a task from the dispatch queue. However, the task could be preempted by other address spaces outside CICS. For example, an address space with a higher dispatching priority could have preempted CICS. Consequently, CICS could be waiting for access to a CPU and not actually executing, although the CICS region would have reported to the Workload Manager that the transaction was in Active state.

Actions to improve performance depend upon whether the server service class (the CICS regions) was denied use of the CPU or whether the server service class was using the CPU[8].

- **Denied use of the CPU**. If the server service class was denied use of the CPU, actions could be taken to increase the relative CPU dispatching priority of the server service class. In Goal Mode, users cannot specify a dispatching priority for address spaces or service classes. The Workload Manager adjusts dispatching priority based upon the importance of performance goals associated with the service class and based on whether the service class is meeting its performance goal. If this service class is denied use of the CPU, consider increasing the Goal Importance of the served transaction service class that was missing its goal.
- **Using the CPU**. If the server service class is primarily using the CPU, actions could be taken to optimize application code of the tasks serving the transactions. These actions should reduce the CPU requirements of the code. Alternatively, performance improvement actions could include increasing the CPU capacity by acquiring a faster processor.

3.3 IDLE STATE.

The idle state indicates that no work requests ready to run in the service class. For CICS transactions, the tasks would be shown as "Suspended" by the CEMT INQUIRE TASK command. This time differs depending upon the types of tasks executing.

- Tasks could be waiting of a principal facility (for example, conversational tasks could be waiting for a resource from a terminal user).
- The Terminal Control task (CSTP) could be waiting for work.
- The inter-region controller task (CSNC) could be waiting for transaction routing requests.
- CICS system tasks (such as CSSY) could be waiting for work.

None of these tasks should be in a service class with a response goal, as neither CICS nor the Workload Manager can provide resources to reduce the response time. Consequently, consider the following alternatives:

- **Modify the workload classification scheme**. The most likely problem is that the workload classification scheme does not adequately partition the transactions between time-critical service classes and service classes that do not have a critical response goal. Consider modifying the workload classification scheme such that the transactions experiencing Idle state time are placed into a service class different from the service class containing important transactions.

  While it may be true that the transactions experiencing Idle state time are "important" transactions, the Workload Manager cannot allocate resources to reduce response for transactions that are in Idle state for reasons outside the Workload Manager's control.

- **Review the performance goal for the service class**. From a "conceptual" view, the transactions experiencing Idle state should be assigned an execution velocity goal; they would receive CPU time when they wanted the CPU time. Unfortunately, the Workload Manager cannot assign resources to transactions, but assigns the resources to address spaces supporting the transactions. Thus, the Workload Manager ISPF application does not allow transaction subsystem service classes to be defined with any goal other than a response goal.

  If a short response goal is specified, the Workload Manager will incur overhead attempting to meet a performance goal for events outside its control. While the Workload Manager will often detect this situation (that is, it will detect that it cannot take action to improve response for the service class), there is no point in having the Workload Manager incur the overhead required to make the decision.

  Consider specifying a **very long** response goal for the service class containing the transactions in Idle state. These transactions are idle (Suspended) waiting for events outside the Workload Manager's control. **This action should be done only after important transactions with valid response goals have been removed from the service class!**

[8] RMF reports detailed information about the “server” service class, which is composed of address spaces (the CICS regions and perhaps IMS regions). See the companion paper “Analyzing RMF Workload Activity Reports in MVS (Goal Mode)” for a discussion of the information available about address spaces.

Modify the workload classification scheme, if necessary, to make sure that the important transactions have been removed from the service class with the long response goal.

3.4 WAIT STATE.

The Wait state indicates that a task in support of the transaction was waiting on some activity. These tasks would be shown as "Suspended" by the CEMT INQUIRE TASK command. The Wait state is broken into eleven categories: Waiting for Lock, Waiting for I/O, Waiting for Conversation, Waiting for Distributed Request, Waiting for a Session to be Established (in the local image, somewhere in the sysplex, or somewhere in the network), Waiting for a Timer, Waiting for Another Product, Waiting for a New Latch, or Waiting for an Unidentified (Miscellaneous) Resource.

With CICS/TS, the CICS Performance Class records produced by the CICS Monitoring Facility (CMF) have been **significantly** enhanced. Of particular interest to performance analysis, the CMF Performance Class records provide a detailed breakdown of transaction suspend (wait) time, reporting 18 categories of wait. This information can be analyzed at the transaction level, or summarized by service class.

Considerable discussion of the Wait state follows, since the Wait state typically offers the most opportunity for improving CICS performance. The discussion is oriented to the CICS Wait State information recorded by RMF for MVS (Goal Mode) environments since many sites are not yet operating CICS/TS[9]. Each possible category of Wait state is discussed below, although a particular Wait state might not affect CICS performance (or might not even apply to CICS). The detailed discussion is provided for completeness.

Table 18 of the referenced *CICS Problem Determination Guides* lists the resources that a suspended task might be waiting on, and shows the WLM Wait Type associated with each wait category.

3.4.1 Waiting for Lock. The Waiting for Lock State indicates that some work request (e.g., a CICS task) was waiting for a lock. There are nine general reasons that CICS provides the Workload Manager with a Wait for Lock[10].

- **Task Control lock 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 a DEQ to release the file. Solving these problems require a review of the application or file design. For example, the task issuing the WAIT EVENT might not have specified the correct event, the task having the lock might be waiting for some event, or the task having the lock might have a performance problem.

- **EXEC CICS ENQ lock waits**. EXEC CICS ENQ lock waits occur when an application issues an EXEC CICS ENQ command to acquire an enqueue on a recoverable resource, and another task already holds an enqueue on the resource. With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for ENQ delays. The CEMT INQUIRE UOWENQ command can be used to discover the owner of the enqueue that the suspended task is waiting on if these lock waits occur.

- **Loader lock 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 is completed, the suspended task is normally resumed quickly and the wait is not likely to be a serious performance constraint. 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 that can be analyzed to decide whether the WLM Lock wait was likely caused by an 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, 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 decide 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 action should be taken.

    If the SMSSOS value is non-zero, review the suggested actions in the referenced *CICS Performance Guides* for reducing the virtual storage requirements above and below the 16MB line.

[9] Additionally, CMF Performance Class records are voluminous and many CICS installations do not collect these records

[10] Table 18 of the CICS *Problem Determination Guide* shows over 30 resources and suspending modules that can report a Wait for Lock to the WLM. This discussion summarizes these into categories of similar Wait for Lock causes.

- **There could be an I/O error on a library**. Check for messages that might indicate an I/O error on a library. If an I/O error occurred, investigate the reason the I/O error occurred.

The CICS Loader Domain Interval Statistics can be analyzed to decide whether loader problems exist. The Loader Domain Statistics report both the number of requests forced to suspend for loader request waits and the total suspend time (fields LDGWTDLR and LDGTTW, respectively).

- **File Control lock waits**. File control lock waits can occur for (1) a record lock in a recoverable VSAM file of a CICS-maintained data table, (2) a record lock in a BDAM file or user-maintained data table, (3) a record lock in VSAM I/O (RLS), (4) a wait for a range lock in a recoverable KSDS data set.

  Depending on the type of file control lock wait, the wait could be very long (for example, if a record update is dependant upon a terminal operator, the file control lock wait could be quite lengthy). CICS interval statistics can be analyzed to assess whether CICS transactions wait for File Control reasons.

- **Temporary storage lock waits**. The most likely causes of temporary storage lock waits occur when (1) a task has issued an auxiliary temporary storage request but the buffers are all in use, (2) a task is waiting for an auxiliary temporary storage VSAM string, (3) a task is forced to wait on TSAUX if it has made an unconditional request for temporary storage and the request cannot be met because insufficient auxiliary storage is available, or (4) a task has issued a request against a temporary storage queue that is already in use by another task[11].

  - The Temporary Storage Statistics in the CICS Interval Statistics can be used to decide whether tasks *waited for temporary storage buffers*. Field TSGBWTN contains the number of times a request was queued because all buffers were allocated to other tasks. Consider increasing the number of auxiliary temporary storage buffers (the TS parameter in the SIT) if significant task delay was caused by all buffers in use.

  - The Temporary Storage Statistics in the CICS Interval Statistics can be used to decide whether tasks *waited for temporary storage strings*. Field TSGVWTN contains the number of times a request was queued because no buffer strings were available. Consider increasing the number of auxiliary temporary storage strings (the TS parameter in the SIT) if significant task delay was caused by all strings in use.

  - The Temporary Storage Statistics in the CICS Interval Statistics can be used to decide whether tasks *waited because auxiliary storage had been exhausted*. Field TSGSTA8F contains the number of times a one or more tasks were suspended because of a NOSPACE condition. Consider increasing the amount of auxiliary temporary storage (or defining secondary extents for the data set) if significant task delay was caused by waiting for auxiliary storage.

  - With CICS/TS, the CMF Performance Class record provides the time transactions *waited for Shared Temporary Storage delays.* Consider creating more temporary storage queues to reduce contention between tasks if shared temporary storage queues significantly delayed tasks. Additionally, examine the tasks using temporary storage queues to see whether unique queue names can be specified or whether the tasks can be modified so they more quickly relinquish the queue.

- **Transient data lock waits**. Transient data lock waits occur when a task is suspended waiting for intrapartition data, waiting for extrapartition data, or waiting because it is trying to read uncommitted data from a logically recoverable transient data queue. Transient data queues are intrapartition if they are associated with a facility allocated to the CICS region, and are extrapartition if the data is directed to a destination that is external to the CICS. The Transient Data Statistics in the CICS Interval Statistics provide information about transient data activity, but do not provide information about waits for either intrapartition or extrapartition destinations.

  With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for Transient data I/O delays.

  - **Waiting for intrapartition data**. The task is suspended waiting for intrapartition data when it has issued a request against an intrapartition transient data queue, but another task is already accessing the same queue. The waiting task cannot resume until that activity is complete.

    If tasks frequently get suspended waiting for intrapartition data locks, determine which other transactions use the same intrapartition destination and consider redefining the intrapartition destinations.

  - **Waiting for extrapartition data**. The task is suspended waiting for extrapartition data when

[11] Table 18 of the *CICS Problem Determination Guide* lists at least seven more causes of temporary storage lock waits. However, these causes either are unlikely to delay user tasks, or they relate to hardware failure or VSAM problems. Consequently, they are not discussed in this document.

it 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. Depending on the system loading, these locks might take several seconds.

If tasks frequently get suspended waiting for extrapartition data locks, determine which other transactions write data to the same extrapartition destination and consider redefining the extrapartition destinations in the DCT (destination control table).

- **Waiting while attempting to read data from a logically recoverable transient data queue**. A task is suspended if it is trying to read uncommitted data from a logically recoverable queue. The task cannot read data that is currently being written to the queue until the task holding the write enqueue commits the changes it has made and dequeues from the write end of the queue.

  Normally, the suspended task should not have a lengthy wait for the data. However, a lengthy wait can occur if the task owning the write enqueue has an indoubt failure, and the queue is defined with the WAIT=YES and WAITACTION=QUEUE attributes. Code NOSUSPEND on the READQ TD request to prevent waiting for data to be committed to the queue.

- **Recovery management lock waits**. Recovery manager lock waits occur only with CICS/TS. Recovery management lock waits occur when the recovery manager is trying to log data for a unit of work while an activity keypoint is moving or deleting the UOW's log data. The suspended task should be resumed when the activity keypoint task finishes working on the log data. Recovery Manager data in the CICS/TS Interval Statistics can be used to decide whether recovery management problems existed.

  With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for Resource Manager Interface delays. If significant Recovery Manager delays occur, try to discover why the activity keypoint task (CSKP) is not completing.

- **Lock Manager lock waits**. The Lock Manager suspends a task when the task cannot acquire the lock on a resource it has requested. A user task cannot explicitly acquire a lock on a resource, but many CICS modules that run on behalf of user tasks do lock resources. Lock Manager waits could indicate a CICS system error.

  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.

  With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for Lock Manager delays. Review the "Lock Manager Waits" part of the Dealing With Waits section of the CICS Problem Determination Guide if significant time was waiting for locks.

- **CICS system task lock waits**. CICS module DFHAPDM is the Application Domain (AP) module responsible for initializing, quiescing, and terminating the application domain. CICS provides the Workload Manger 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.

3.4.2 Waiting for I/O. The Waiting for I/O state indicates that the subsystem (the CICS or IMS region) was waiting for some I/O request on behalf of the served transaction service class. This state could be waiting on an actual I/O operation or waiting on another function related to the I/O request. These tasks would be shown as "Suspended" by the CEMT INQUIRE TASK command.

The Waiting for I/O is not necessarily time actually performing I/O, but could be any activity related to the I/O request. For CICS transactions, this time includes:

- File Control requests.
- Transient data requests.
- Temporary storage requests.
- Journaling I/O requests.
- Waiting for I/O buffers or VSAM strings.

CICS Interval Statistics can be used to identify causes of the Wait for I/O problems.

With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for I/O, for each CICS transaction that was Suspended for I/O. Ten categories of Wait for I/O are provided (Terminal I/O wait time, File I/O wait time, RLS File I/O wait time, Journal I/O wait time, Temporary Storage I/O wait time, Shared Temporary Storage I/O wait time, Inter-Region I/O wait time, Transient Data I/O wait time, LU 6.1 I/O wait time, and LU 6.2 I/O wait time).

3.4.3 Waiting for Conversation. The Waiting for Conversation state indicates that the subsystem was waiting for a response in a conversation mode. The

percent spent in the Wait for Conversation state should be approximately the same as the sum of the Switched states.

3.4.4 Waiting for Distributed Request. The Waiting for Distributed Request state indicates that some function or data must be routed before resumption of the work request. The distributed request function is not used by CICS, so values should never appear in this wait state.

3.4.5 Waiting for Session to be Established (established locally in the current MVS image, established in the sysplex, or established in the network). Values in these wait states should not appear except during intervals when CICS regions are started (and when the sessions are established). Sessions are normally established for prolonged periods. If a transaction service class waits for a session to be established for a production environment, there may be operational problems or CICS region integrity problems.

3.4.6 Waiting for timer. The Waiting for Timer state means that a work request was waiting for expiration of a timer. Consider the following alternatives if significant time was spent in this state:

- Identify the transactions that cause the Wait for Timer delay. Consider placing these transactions into their own service class, as it is usually inappropriate for transactions that wait for a timer to be in a service class with time-critical transactions.

- Alternatively, review the performance goal associated with these transactions. It is possible that the transactions have been placed into their own service class, but the performance goal associated with the service class does not adequately account for the timer delays. Since timer delays are typically an application-related function, the performance goal may need to be revised to account for longer delays.

- Alternatively, the applications may have issued a timer delay because of the unavailability of some CICS resource. Review the application to decide the cause of the timer delay and whether the delay can be reduced.

3.4.7 Waiting for Another Product. The Waiting for Another Product means that a work request was waiting for another product to provide some service. For CICS, these products typically will be DBCTL or DB2. If the delay is significant, the performance goals for the service class may be achieved only if the performance of the other product can be improved.

3.4.8 Waiting for a New Latch. The Waiting for a New Latch means that a work request was waiting for a new latch. A latch is a short-duration lock, and CICS does not directly report this state.

3.4.9 Waiting for Unidentified Resource. The Waiting for Unidentified Resource means that the work request was waiting, but that the subsystem could not identify the cause of the wait. These waits are categorized as Miscellaneous Waits. There are twelve general reasons that CICS provides the Workload Manager with a Miscellaneous Wait[12].

- **CICS system task waits**. CICS system task waits occur (1) as a natural result of the CICS system tasks or (2) because of a system error preventing the system task from resuming.

  - Many system tasks enter a wait state as a natural result of their operation. For example, the DFHSMSY module of the storage manager domain might stay suspended for a prolonged time (i.e., minutes, or even hours). The purpose of the DFHSMSY module is to clean up storage when significant changes occur in the amount being used. This situation should happen infrequently in a production system.

  - Some system tasks perform many I/O operations. These I/O operations are subject to I/O constraints such as string availability, and volume and data set locking. With tape volumes, the tasks can also be dependent on operator action while new volumes are mounted.

  Consider placing CICS system tasks into a single service class. IBM suggests that CICS-supplied transactions not be placed in a service class with user transactions. Contact an IBM support center if a system task is in a wait state, and there is a system error preventing it from resuming.

- **Execution Diagnostic Facility (EDF) waits**. The EDF waits are a natural result of using the Execution Diagnostic Facility. The EDF waits should not occur in a CICS production region. EDF waits would not be a cause for concern in a CICS test region, as they are programmer-generated.

- **Front End Programming waits**. There are two types of Front End Programming waits from the view of CICS: (1) a wait for the FEPI_RQE resource and (2) a wait for the SCRDP resource.

  - The wait for the FEPI_RQE resource is issued in the FEPI adapter when a FEPI command is passed to the Resource Manager for processing. The wait ends when the Resource Manager has processed the request. Being outstanding for a long time is possible for a FEPI_RQE wait (for

[12] Table 18 of the CICS *Problem Determination Guide* shows over 60 resources and suspending modules that can report a Wait for Miscellaneous reasons to the WLM. This discussion summarizes these into categories of similar Miscellaneous Wait causes.

example, when awaiting a flow from the back-end system that is delayed due to network traffic).

- The wait for the SCRDP resource is issued by the CSZI task in the FEPI Resource Manager when it has no work to do. The wait ends when work arrives (from either the FEPI adapter or a VTAM exit). An SZRDP wait is generated when the FEPI Resource Manager is idle. Consequently, the SZ TCB is also inactive. On lightly loaded systems, this wait occurs frequently, but would not cause a transaction service class to miss its performance goal.

The Dispatcher Domain Statistics part of the CICS interval statistics contain information that can be analyzed to decide whether the WLM Miscellaneous Wait was likely caused by a Front End Programming wait. There are Dispatcher Domain Statistics for each TCB; TCB 4 is the secondary LU TCB and is present if FEPI=YES was specified in the System Initialization Table. Within TCB 4 statistics, the DSGTWT field holds the accumulated real time that the CICS region was in a MVS wait for the Front End Programming TCB.

If the DSGTWT value is small, the WLM Miscellaneous waits probably were **not** caused by Front End Programming waits.

If the DSGTWT value is relatively large, it is possible that the WLM Miscellaneous waits **were** caused by Front End Programming waits. Unfortunately, there is no way to decide whether a task suspended for a Front End Programming Wait actually was in the service class missing its performance goal. However, **some** tasks in the CICS region are encountering Front End Programming Waits if the DSGTWT value is relatively large and action should be taken to reduce the wait.

The CICS Front End Programming Interface User Guide (see References) should be consulted regarding improving the performance of the Front End Programming interface.

Additionally, consider placing CICS system tasks into a single service class. IBM suggests that CICS-supplied transactions not be placed in a service class with user transactions.

- **Interval Control waits**. Interval Control waits are caused by user tasks. The "Interval Control Waits" part of the Dealing With Waits section of the referenced *Problem Determination Guides* should be reviewed to select ways for reducing these waits.

- **Journal Control waits**. CICS Journal Control provides the Workload Manager with a Miscellaneous Wait for four resource types: JASUBTAS, JCBUFFER, JCDETACH, and JCREADY.

  - **JASUBTAS**. The purpose of the wait for the JASUBTAS resource is to delay shutdown until the JASP subtask has completely submitted all the archiving jobs of those journals needing to be archived.

  - **JCBUFFER**. If the resource type is JCBUFFER, with resource name JCTBAECB, the task that has requested shutdown is waiting for the Journaling task to flush the buffer, close the journal, and terminate itself.

  - **JCDETACH**: A task that has requested shutdown can be made to wait on the detaching of the journal subtask from the operating system.

  - **JCREADY**. Workload Manager Miscellaneous Waits for the JCREADY resource type occur during archiving. CICS writes to a second data set while archiving the first data set either tape or disk. The first data set is not reused until archiving is complete and the operator has responded to message DFHJC4583. If the operator has not responded before the second journal data set is full, the JCT PAUSE option causes logging to cease until the operator has responded. User tasks are made to wait on resource type JCREADY when no operator reply has been received to message DFHJC4583, and message DFHJC4584 has subsequently been issued.

  Workload Manager Miscellaneous Waits for the first three Journal Control resource types occur only during shutdown, and should not cause a service class to miss its performance goal.

  Workload Manager Miscellaneous Waits for the JCREADY resource type could cause serious performance problems if the operator does not quickly respond to message DFHJC4583.

  The Journal Control Statistics part of the CICS interval statistics contains information that can be analyzed to decide whether the WLM Miscellaneous Wait was likely caused by CICS having to wait for the archive job. The field A13WAC is a count of the number of times CICS had to wait for a particular journal because the archive job had not completed at the time it was needed.

  - If the A13WAC field is zero, the WLM Miscellaneous waits were **not** caused by Journal Control archiving.

- If the A13WAC value is non-zero, determine why a quick response was not provided for message DFHJC4583. While it is uncertain that the operator's response caused problems with the service class missing its performance goal, tasks are suspended because of archiving problems. Action should be taken to correct the problem.

- **Storage waits**. Storage waits occur when a task is waiting for any of the resource types CDSA, UDSA, ECDSA, EUDSA, ERDSA, SDSA, ESDSA, or RDSA. Waits on these resources occur when tasks make unconditional storage requests (SUSPEND=YES) that cannot be satisfied[13]. Storage requests below the 16MB line wait for CDSA, UDSA, SDSA, or RDSA. Storage requests above the line 16MB line wait for ECDSA, EUDSA, ESDSA, or ERDSA.

  The most likely reasons for extended waits on storage requests are:

  - The task has issued an unconditional GETMAIN request for an unreasonably large amount of storage.

  - The task has issued an unconditional GETMAIN request for a reasonable amount of storage, but the CICS region is approaching a short-on-storage (SOS) condition.

  - The task has issued an unconditional GETMAIN request for a reasonable amount of storage, but storage in the CICS region could have become too fragmented for the request to be satisfied.

  The Storage Manager Statistics part of the CICS interval statistics contain information that can be analyzed to decide whether the WLM Miscellaneous Wait was likely caused by a storage wait. The field SMSUCSS is a count of the number of times a task was suspended because of insufficient storage to satisfy the request.

  - If the SMSUCSS value is zero, the WLM Miscellaneous waits were **not** caused by storage waits.

  - If the SMSUCSS value is non-zero, it is possible that the WLM MISCELLANEOUS waits **were** caused by storage waits. Unfortunately, there is no way to decide whether a task suspended for storage constraint actually was in the service class missing its performance goal. However, tasks in the CICS region are encountering waits for storage if the SMSUCSS value is non-zero, and action should normally be considered. Further, the waiting task may be automatically purged[14] if it has waited for storage longer than the deadlock time-out parameter specified in the installed transaction definition.

    If the SMSUCSS value is non-zero, review the checklist for reducing the virtual storage requirements above and below the 16MB line, contained in the referenced *CICS Performance Guides.*

- **Task Control waits**. The CICS Transaction Manager provides the Workload Manager with a Miscellaneous Wait when a task is waiting on a resource type of KCCOMPAT, and the task has been suspended by the Transaction Manager. Additionally, CICS Task Control provides the Workload Manager with a Miscellaneous Wait when a task is waiting on a resource type of EKCWAIT and has been suspended by Task Control.

  Task Control waits tend to be application-dependent. Review the "Task Control Waits" part of the Dealing With Waits section of the referenced *Problem Determination Guides.*

- **Temporary Storage Waits**. Temporary storage is a scratchpad facility that is heavily used in many systems. Temporary storage exists in either main storage above the 16MB line (ECDSA), or auxiliary storage in a VSAM-managed data set. Temporary storage waits are related to temporary storage existing in auxiliary storage.

  The Temporary Storage Statistics part of the CICS interval statistics contain information that can be analyzed to decide whether the WLM Miscellaneous Wait was likely caused by a Temporary Storage wait. The field A12STA8F field is a count of the number of times a task was suspended or had been abended because auxiliary storage had been exhausted.

  - If the A12STA8F value is zero, the WLM Miscellaneous waits were **not** caused by Temporary Storage waits.

  - If the A12STA8F value is non-zero, it is possible that the WLM Miscellaneous waits **were** caused by Temporary Storage waits. Unfortunately, CICS Interval Statistics do not contain information to decide whether a task suspended for Temporary Storage constraint actually was in

[13] Note that, if conditional requests are made (SUSPEND=NO), tasks are not suspended on these resources, and a miscellaneous wait would not be provided to the Workload Manager.

[14] Certain conditions prevent purging of a task (as examples, a deadlock time-out value of 0, or a specification of SPURGE(NO)).

the service class missing its performance goal[15]. However, tasks in the CICS region are encountering waits for Temporary Storage if the A12STA8F value is non-zero, and action should be taken. Further, the waiting task may be automatically purged[16] if it has waited for temporary storage longer than the deadlock time-out parameter specified in the installed transaction definition. Otherwise, the waiting task is not purged, and it could be suspended indefinitely.

If the A12STA8F value is non-zero, review the checklist for improving the performance of temporary storage residing on auxiliary storage, contained in the IBM *CICS Performance Guide*.

- **Transient Data waits**. Workload Manager Miscellaneous Waits for Transient Data occur only during system initialization. These waits would not cause a service class to miss its performance goal because the region has not yet begun accepting transactions.

- **User waits**. CICS provides the Workload Manager with a Miscellaneous Wait when a task is waiting on an ECB list posted by the user. User waits are application dependent.

- **VTAM waits**. CICS provides the Workload Manager with a Miscellaneous Wait when a task is waiting on three resource types: ZCIOWAIT, ZCZGET, and ZCZNAC.

  - The ZCIOWAIT resource type wait is caused by a task waiting on terminal I/O.

  - The ZCZGET resource type wait is caused with application request logic for LU6.2 devices.

  - The ZCZNAC resource type wait is for DFHZNAC to issue an error message.

- **XRF alternate system waits**. CICS provides the Workload Manager with a Miscellaneous Wait when a task is waiting caused by XRF alternative system waits. The XRF takeover process is a major system event, and individual tasks would not to perform well during the takeover.

To summarize the above discussion, the most likely causes of Workload Manager Miscellaneous Waits, during **normal** transaction processing, are: (1) CICS system task waits, (2) storage waits, (3) temporary storage waits, and (4) application-dependent waits.

- Consider placing CICS system tasks into a single service class. IBM suggests that CICS-supplied transactions not be placed in a service class with user transactions. Once CICS-supplied transactions have been removed from the service class missing its performance goal, remaining waits would be related to SUSPENDED user tasks.

- Examine CICS interval statistics to decide whether the Miscellaneous Waits are related to storage waits or temporary storage waits. The preceding discussion describes the relevant fields in the interval statistics.

- With CICS/TS, examine CMF Performance Class records to decide whether the Miscellaneous Waits are related to storage waits or temporary storage waits.

- If the actions suggested in the preceding sections have been taken and Miscellaneous Waits remain a major cause of transaction delay during normal operations, the most likely cause is application-dependent waits. Examine applications to decide whether they cause the waits, or simply ignore the waits.

3.5 SWITCHED STATE. The Switched state indicates that processing of the transaction had been switched from the CICS or IMS subsystem providing information to the Workload Manager. The transaction could have been switched to another CICS region (for example) in the same MVS image, switched to another MVS image in the sysplex, or switched somewhere in the network. Detail about the state of a transaction in the Execution Phase depends upon where the transaction is switched.

- **Switched in the local MVS image**. If the transaction is switched in the local MVS image to another CICS region, the transaction state information for the various CICS regions is combined in the "CICS subsystem" information in the SMF Type 72 records. If the transaction is shipped to an IMS region (perhaps for data access), the IMS subsystem information is provided separately under the "IMS subsystem" information in the SMF Type 72 records.

- **Switched in the sysplex**. When the transaction is switched to another MVS image in the sysplex, the receiving subsystem on the new MVS image provides transaction delay information to the Workload Manager (provided as Execution Phase data). At present, analyzing delays is difficult once a transaction has been switched to another system in the sysplex. Several scenarios complicate the analysis:

[15] With CICS/TS, the CMF Performance Class record provides the time transactions were waiting for temporary storages

[16] Certain conditions prevent purging of a task (as examples, a deadlock time-out value of 0, or a specification of SPURGE(NO)).

- The sysplex is set up in a "standard" way in which a CICS Terminal Owning Region (TOR) is started in one system and CICSplex/SM is used to switch transactions to Application Owning Regions (AORs) on a number of systems. This is the simplest to evaluate, as there is some correlation between BTE Phase in the TOR system and Execution Phase time in the other systems.

- The sysplex is set up with TORs on more than one system, transactions can be submitted to the different TORs on different systems, and the transactions are switched among systems on the sysplex. It becomes unclear which system actually processes the transactions of a transaction service class missing its performance goal. (That is, the transactions might process satisfactorily on one system but not perform well on another system.)

- **Switched in the network**. If the transaction is switched somewhere in the network. the Workload Manager has no more information about the status of the transaction; it is simply "switched in the network" from the Workload Manager's view.

4.0 CONCLUSION

With the data available in MVS (Goal Mode), IBM has provided a wealth of information about CICS performance and performance constraints. This information ranges from the basic data available from RMF about transaction performance, to the information provided to the Workload Manager by CICS (and reported by RMF) describing the transactions' state as each transaction is processed, to the detailed CMF Performance Class statistics provided by CICS/TS describing specific delays to transactions.

The IBM CICS Development Team has been extremely receptive to providing information needed to perform a comprehensive analysis of CICS.

5.0 REFERENCES

CICS/ESA Version 4.1 Performance Guide
Section 2.7.1.1: Response time breakdown

CICS/ESA Version 4.1 Problem Determination Guide
Section 2.3: Dealing with waits

CICS/ESA Front End Programming Interface User Guide
Section 2.4.2 (System-related performance)
Section 3.4.5.2 (Application-related performance)

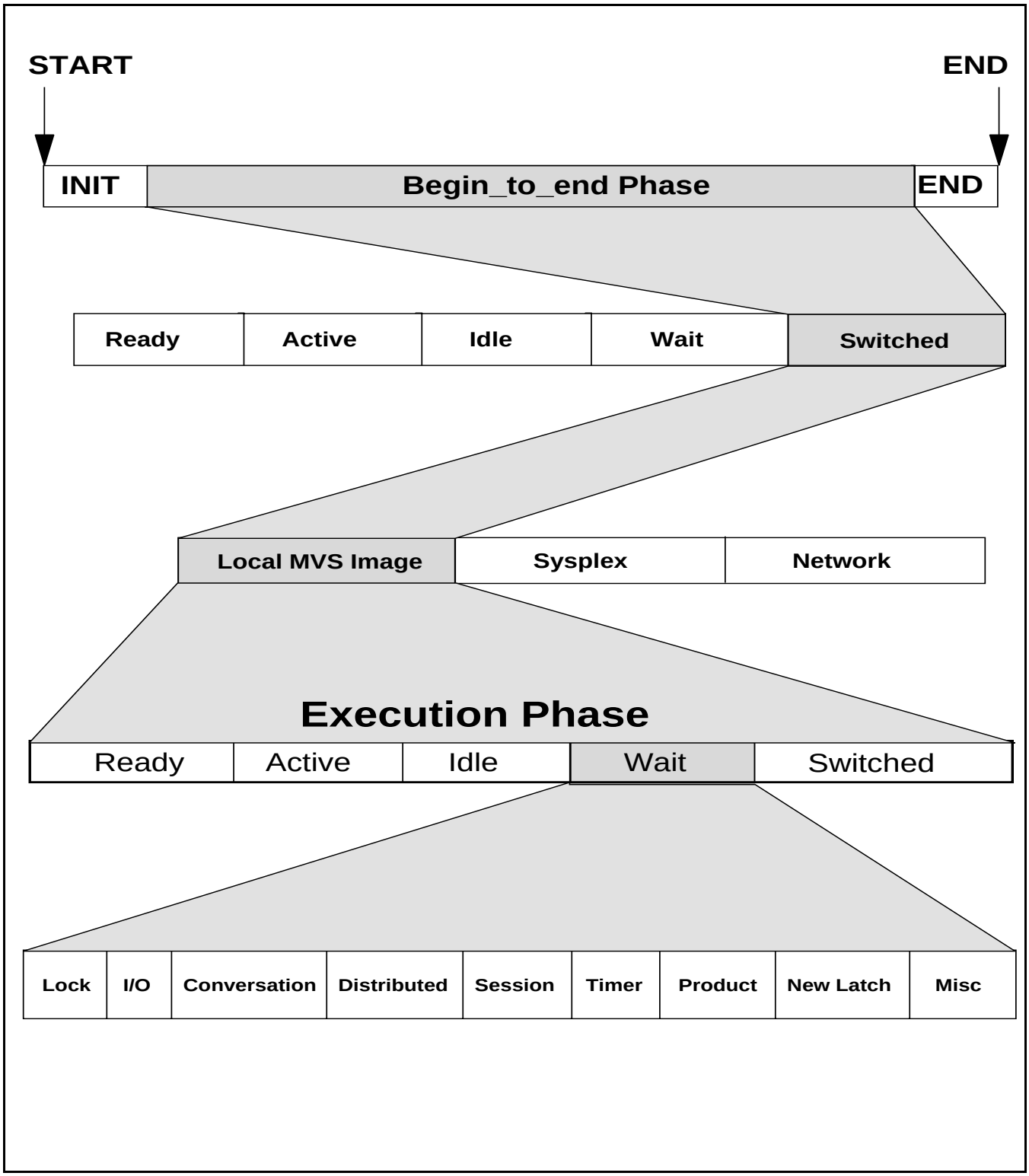
CICS/TS Release 1.2 Performance Guide
Section 2.7.1.1: Response time breakdown

CICS/TS Release 1.2 Problem Determination Guide
Section 2.3: Dealing with waits

CICS/TS Front End Programming Interface User Guide
Section 2.4.2 (System-related performance)
Section 3.4.5.2 (Application-related performance)

6.0 ACKNOWLEDGMENTS

The author would like to recognize the efforts of the IBM CICS/ESA Development Team, IBM United Kingdom Laboratories (particularly Mr. Chris Baker) for providing detailed information about the resources on which a CICS task might be waiting. Based on an informal request to Chris at the August 1995 SHARE Technical Conference, IBM revised its *CICS Problem Determination Guides* to include a detailed itemization of the CICS waits. This invaluable information allows performance analysts to provide a more comprehensive analysis of CICS delays.



Note: Multiple execution phases may exist.

WLM VIEW OF CICS TRANSACTIONS

Figure 1