Rule WLM103: Service Class did not achieve execution velocity goal

Finding: CPExpert has detected that a service class period did not achieve the execution velocity goal that was specified in the Service Policy in effect.

Impact: This finding can have a HIGH IMPACT on performance of your computer system.

Logic flow: This is a basic finding. There are no predecessor rules.

Discussion: Installations may specify an *execution velocity goal* for a service class period. An execution velocity is a measure of how fast work should run when the work is ready to run, without being delayed waiting for access to a CPU or delayed waiting for access to processor storage[1]. The purpose of specifying an execution velocity goal is to allow installations to specify how important it is to have work processed, when the work has no time-related measure (that is, a response requirement is not associated with the work).

With OS/390[2], the execution velocity is computed based on samples collected at periodic sampling intervals[3] by the System Resources Manager (SRM). The SRM sampling code interrogates address space control blocks (TCBs, SRBs, OUCBs, and OUXBs) to determine the state of each address space assigned to a service class. Sampling counts associated with the service class are updated based upon the state[4] of the address spaces.

The sampling code records the sampling result into the following categories:

- **CPU using samples**. CPU using samples mean that the address space is using the CPU.

[1]Processor storage is composed of *central storage* and *expanded storage*. The third category of storage is *auxiliary storage*.

[2]With z/OS, SRM uses "derived" samples for CPU using samples. This "derived" sample approach is described later in this document..

[3]With MVS/ESA SP5.1, the sampling interval is 250 milliseconds. The state of each TCB or SRB associated with an address space is sampled every 250 milliseconds, beginning from address space initiation.

[4]Note that an address space can be in multiple states (for example, a CICS region might be using multiple processors concurrently, while some CICS tasks were also waiting on some function). Thus, the sample counts can total more than 100% of the sample intervals for any address space.

- **I/O using samples**. I/O using samples means the number of calculated samples of work using non-paging DASD I/O resources (DASD connect state or DASD disconnect state[5]). I/O using samples are included only if the installation has elected to include WLM-managed I/O.

  For most samples that are taken by the WLM, the WLM can sample dispatchable units to see what state they are in (they are using the CPU, or they are delayed for specific reasons). At each sampling interval, the WLM simply examines the state of the dispatchable unit and adds a count of "1" to the appropriate counter reflecting the state of the dispatchable unit.

  This sampling approach cannot be used with DASD I/O operations, because the DASD values are not available to WLM as instantaneous "states," a state sampling approach cannot be used. DASD I/O time is reported to MVS as counters accumulated by the I/O controllers.

  Consequently, the WLM *calculates* the number of samples of work using non-paging DASD I/O resources. The WLM uses the device connect time (and device disconnect time if APAR OW47667 is *not* installed or with z/OS V1R3) to yield device using time. The WLM multiplies that *time* by the "WLM sampling rate" of 4 samples per second.

  For example, assume a DASD non-paging device using time of 5 seconds accumulated in the previous WLM 10-second policy adjustment interval. The WLM would add 20 I/O using samples for the 10-second policy adjustment interval.

$$I/O\ using\ samples = device\ use\ time * samples\ second$$

$$I/O\ using\ samples = 5\ seconds * 4\ samples\ second = 20\ samples$$

- **CPU delay samples**. CPU delay samples mean that the address space is ready to use the CPU but is being delayed. Two separate CPU delays are recorded:

- **CPU delay**. CPU delay means that a TCB or SRB is waiting to be dispatched or a TCB is waiting for a local lock. CPExpert refers to this delay as "DENIED CPU" in various reports resulting from the analysis of Workload Manager constraints.

[5]With APAR OW47667 (and included in z/OS V1R3), disconnect time is no longer counted as productive I/O time. Disconnect time also is not counted as I/O delay because there is nothing WLM can do to reduce disconnect time.

  - **CPU Capping delay**. This delay to response time means that the maximum CPU service units had been consumed for the Resource Group to which the service class was assigned, and the Workload Manager had marked all address spaces associated with the Resource Group as non-dispatchable for some time-slice intervals.

    This delay does not necessarily mean that address spaces in the capped service class had consumed the CPU service units. The CPU service units could have been used by another service class if more than one service class had been assigned to the Resource Group.

- **Processor storage delay**. Processor storage delay samples means that an address space is ready to execute, but is delayed waiting for processor storage. Eight separate processor storage delays are recorded:

  - **Swap-in delay**. Swap-in delay means that the address space was delayed on swap-in (the swap-in had started, but had not completed).

  - **MPL delay**. MPL delay means that an address space was ready to be swapped in, but that the SRM had not initiated a swap-in because of target MPL constraints.

  - **Auxiliary page delay from private**. This page-in delay means that the address space experienced a page fault in the private area and the pages were coming from auxiliary storage.

  - **Auxiliary page delay from common**. This page-in delay means that the address space experienced a page fault in the Common area and the pages were coming from auxiliary storage.

  - **Auxiliary page delay from cross memory**. This page-in delay means that the address space experienced a page fault from cross memory and the pages were coming from auxiliary storage.

  - **Auxiliary page delay from VIO**. This page-in delay means that the address space experienced a page fault in VIO and the pages were coming from auxiliary storage.

  - **Auxiliary page delay from standard hiperspace**. This page-in delay means that the address space experienced a page fault from standard hiperspace and the pages were coming from auxiliary storage.

  - **Auxiliary page delay from ESO hiperspace**. IBM has defined this state to mean that the address space was experiencing page faults in ESO hiperspace and the pages were coming from auxiliary storage. Pages in ESO hiperspace are, by definition, resident only in expanded storage (ESO = Expanded Storage Only), and are never migrated to auxiliary storage. IBM offers the following explanation[6]:

"The execution delay for ESO hiperspaces is a calculated value based on the assumption that if an application does a read for an ESO hiperspace page and that page is no longer available (has been cast out), the application will read the data from DASD somewhere. WLM/SRM takes the number of times a read failed in this way and multiplies it by the number of delay samples we expect a read of a page from DASD to represent and report the product as the execution delay samples for ESO hiperspace. This obviously is not a perfect solution, but we needed some way to get an estimate of how much delay is caused to an address space by not having enough expanded for an ESO hiperspace. Such an estimated is needed to properly manage the amount of expanded owned by the address space to the address space's goal."

- **Shared page-in delay from auxiliary storage**. This page-in delay means that the address space experienced page faults from shared pages and the pages were coming from auxiliary storage.

- **Shared page-in delay from expanded storage**. This page-in delay means that the address space experienced page faults from shared pages and the pages were coming from expanded storage.

- **Non-paging DASD I/O operations**. With OS/390 Release 3, execution velocity can optionally include delays waiting for non-paging DASD I/O operations. Non-paging DASD I/O delays include IOS queue delays, subchannel pending delays, and control unit queue delays. Note that DASD disconnect time is not included in the execution velocity delay calculations, but could be included in the “using” component of the calculation. *See Footnote 1.*

- **Delays waiting for an initiator**. With OS/390 Version 2 Release 4, execution velocity can optionally include delays waiting for an initiator (with batch jobs in WLM-managed job classes).

Notice that only certain delay categories are included: only delays for processor or for processor storage are included in the "delay" category. These delays are under control of the SRM. Delays not under control of the SRM are not included in CPU or processor storage delays, but are

[6]IBM TALKLink RMF FORUM appended at 15:39:18 on 95/05/29 GMT (by YOCOM at KGNVMC) Subject: Workload Activity Report. Used with permission of the author.

included in an "unknown" delay category. **Unknown delay is not included in the execution velocity computation**.

For example, delay waiting for ENQ completion is not under control of the SRM. Consequently, the Workload Manager does not include waiting for ENQ completion in when it computes execution velocity. Rather, waiting for ENQ completion is included in an "unknown" category when the SRM takes its samples. The "unknown" delay means that the SRM was unable to identify the cause of delay. In practice, this means that the delay was something over which the SRM had no control (e.g., certain I/O operations, ENQ delay, etc.).

The Workload Manager computes the execution velocity of a service class by applying the following algorithm:

$$\frac{\textit{using samples}}{\textit{using samples} + \textit{delay samples}} * 100$$

where:

using samples include:

- The number of samples of work using the processor (CPU Using).
- The number of calculated samples of work using non-paging DASD I/O resources (DASD connect state or DASD disconnect state). I/O using samples are included only if the installation has elected to include WLM-managed I/O. DASD disconnect is not used with APAR OW47667 (and included in z/OS V1R3).

delay samples include:

- The number of samples of work delayed for the processor (Denied CPU Delay or CPU Capping delay).
- The number of samples of work delayed for processor storage. Delay for processor storage includes:
    - Paging delay
    - Swap-in delay
    - Swapped out for multiprogramming (MPL) reasons
    - Server address space creation delay

    - Initiation delays for batch jobs in WLM-managed job classes

- The number of calculated samples of work delayed for non-paging DASD I/O resources (DASD IOS queue delay, DASD subchannel pending delay, or DASD control unit queue delay). I/O delay samples are included only if the installation has elected to include WLM-managed I/O.

The result from the algorithm is multiplied by 100, to yield an execution velocity ranging from 0 (when the address space did not use the CPU) to 100 (when the address space was not delayed for any reason controlled by the SRM).

As mentioned in an earlier footnote, the way that SRM acquires CPU Busy samples changed with z/OS. This change was caused by CPU Busy sampling problems in a PR/SM environment. These sampling problems became acute in a Intelligent Resources Director (IRD) environment.

- Execution velocity had always been potentially incorrect in a PR/SM environment. The SRM samples were based on a 250 millisecond timer interrupt, based on the Time-of-Day Clock. This approach works very well in a non-PR/SM environment but might not work so well in a PR/SM environment. Execution velocity can potentially be SERIOUSLY incorrect in a heavily loaded Central Processor Complex (CPC), if there is a high logical-to-physical processor ratio. In such an environment, PR/SM frequently intercepts logical processors because the PR/SM dispatch interval has lapsed.

- When PR/SM intercepts a logical processor because the PR/SM dispatch interval is exceeded, MVS doesn't “know” about this interception. PR/SM does store various clocks (such as the CPU Clock) when it intercepts a logical processor. Consequently, actual CPU Busy times are not incorrect if the logical processor is intercepted. However, the Time-of-Day Clock cannot be stored and retrieved, because time of day is time of day; time of day continues to tick along regardless of whether the logical processor is dispatched to a physical processor or the logical processor is on the PR/SM Logical Processor Ready Queue[7].

- When an intercepted logical processor is again dispatched to a physical processor, time had passed. As mentioned, MVS takes its samples based on the 250 millisecond timer interrupt (and the timer interrupt is based on time of day). Since time has lapsed while MVS was not in control, the interrupt would not be 250 milliseconds from the perspective

[7] Please refer to the documentation for Rule WLM820 for a discussion or PR/SM concepts, including the Logical Processor Ready Queue.

of MVS. Consequently, samples of CPU Busy[8] could cover a shorter interval from the perspective of MVS, and the count of CPU Busy samples could be higher than might be expected. Since the CPU would be busy when PR/SM intercepted the logical processor, MVS would more frequently find the CPU to be busy when it took its samples based on lapsed time.

- The lapsed time included time when the logical processor was not dispatched to a physical processor, but SRM did not know this and thus counted a sample of CPU busy assuming that it was 250 milliseconds of lapsed time, when it might be only 200 milliseconds or 100 milliseconds, or any random time value less than 250 milliseconds. This anomaly caused the execution velocity to be overstated, by the ratio of MVS CPU Busy to Actual CPU Busy.

- This overstatement did not cause much consternation, until IRD was introduced. IRD bases its decisions on whether a service class was delayed based on CPU access. The IRD decisions could be seriously wrong since the execution velocity could be significantly overstated!

IBM decided to compute the CPU Busy samples based on measured CPU Busy time (since this CPU Busy time was measured by the CPU Clock). The computation recognized that if there was one second of measured CPU Busy, then SRM would have taken four CPU Busy samples during that one second had it sampled once per 250 milliseconds. Consequently, the algorithm simply multiplies the number of seconds of CPU Busy by four, to "derive" the number of SRM samples that would have been taken.

This decision was embodied as a part of changes introduced with APAR **OW47277**. Here is a brief discussion from OW47277 that implemented this change (and many other changes):

> <from APAR OW47277 Problem Description>
> "13. In LPAR mode, there can be "phantom" CPU using samples for a service class when the logical CPU is dispatched but not getting access to a physical CPU. This could distort WLM's decisions by making it look like the work is getting better access to the CPU than it really is."
>
> <snip from APAR OW47277 Problem Conclusion>
> "13. Make the CPU using samples more accurate, especially in an LPAR environment, by deriving CPU using samples from CPU service time rather than from direct sampling. This change may result in lower CPU using samples and

[8] (PR/SM intercept happens only with CPU Busy state, because otherwise MVS would be waiting and would voluntarily give up the physical processor, assuming that WAIT COMPLETION=YES was not specified).

therefore may affect achieved velocities for systems running in LPAR mode. Achieved velocities will be the same or lower, depending on if there is a change in CPU using samples. Customers should review velocity goals in their WLM policy and adjust downward if needed. This change has been incorporated into base z/OS R1.2 and above. "

- IBM knew that in a heavily-loaded PR/SM environment with a large MVS CPU Busy to CPU Busy ratio, there could be a large difference between derived CPU Busy samples versus Sampled CPU Busy counts. For example, if CPU Busy was 30% and MVS CPU Busy was 60%, there would be approximately twice as many CPU Busy Samples as there would be derived CPU Busy Sample counts. The question was: what to do with the excess samples? Clearly, the work was being delayed because it was dispatched to a logical processor but the logical processor was not dispatched to a physical processor.

- With APAR OW47277, IBM decided to put the difference between derived CPU Busy samples and Sampled CPU Busy samples into CPU Delay samples. This algorithm had two effects: using derived CPU Busy samples decreased the value in the numerator, and counting the difference in sample count between the two methods as CPU Delay caused the denominator to be increased. The result could be a **significantly** decreased execution velocity when compared with execution velocity before APAR OW47277.

- This decision (putting excess samples into CPU Delay) caused other problems with Policy Adjustment analysis based on CPU Delay, both at the local system and at the IRD level. Consequently, APAR **OW55665** was issued to correct the problems. With APAR 0W55665, the "excess" samples were not put into CPU Delay, but were ignored.

  Additionally, if the conversion of service times to samples resulted in a fraction, the service time that could not be converted to samples was accumulated in a control block associated with the service class period. This accumulation would (1) either grow large enough to be converted to samples, or (2) would remain small with the inference that there really was nothing to worry about.

The samples that are in SMF Type 72 (Subtype 3) records are the values that result from either the SRM sampling once per 250 milliseconds with OS/390, or from the "derived" samples based on measured CPU Busy with z/OS. The ultimate calculation of execution velocity (and calculation of Performance Index) is unchanged from the discussion presented earlier.

It is important to keep in mind that execution velocity applies **only to times when an address space is using a CPU or ready to use a CPU (or**

using I/O or ready to use I/O if WLM-managed I/O is included). It does not include times when an address space is idle, waiting for I/O (if WLM-managed I/O is not included), enqueued for a resource, etc.

The Workload Manager periodically[9] computes the execution velocity of all address spaces that have an execution velocity goal.

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. This assessment is referred to as the "policy adjustment" interval, in that the Workload Manager decides whether to adjust resource policies based on whether service classes are meeting performance goals.

The actual comparison process is accomplished by computing a *Performance Index* for each service class[10]. For execution velocity goals, the performance index is computed by dividing the goal by the achieved velocity. If achieved velocity is greater than the goal, the performance index will be less than one. If achieved velocity is less than the goal, the performance index will be greater than one.

- For example, suppose that an execution goal of 30% had been specified. Further suppose that the execution velocity achieved was 50%. Dividing the goal by the achieved would yield a performance index of 0.6 (30%/50%=0.6).

- However, suppose that the execution velocity achieved was only 15%. Dividing the goal by the achieved would yield a performance index of 2.0 (30%/15%=2.0).

As can be seen by the above discussion, a performance index less than one implies that a performance goal **has** been met, while a performance index greater than one implies that a goal has **not** been. Thus, the performance index can be used to compare the performance of service classes, regardless of the type of performance goal specified for the service class[11].

For service classes that have an execution velocity goal, the Workload Manager determines whether the execution velocity is less than the performance goal. If the execution velocity is less than the performance goal, the system is not meeting performance goals for the service class

[9]The Workload Manager computes the execution velocity every 10 seconds, during the "policy evaluation" interval.

[10]Please see Section 4 for a discussion of how the Performance Index is computed and used.

[11]A discretionary goal has an implied performance index of 81%, which means that service classes with discretionary goals will always be considered as achieving their service goal.

period. If the importance of the service class is sufficiently high, the Workload Manager may reallocate system resources in an attempt to meet performance goals.

At a different period (typically every 15 minutes), the SRM provides RMF with measurement data, including the CPU Using, CPU Delay, and Storage Delay samples for each service class period. This information is collected by RMF and written to the SMF data set as Type 72 records. The interval in which RMF collects data and writes records typically is referred to as the *RMF measurement interval*.

CPExpert analyzes the SMF Type 72 records to determine whether service class periods met their performance goals during each RMF measurement interval. For service class periods that have an execution velocity performance goal specified, CPExpert accomplishes this simply by dividing the CPU Using samples (R723CCUS) by the total Using and Delay samples (R723CCUS + R723CTOT). The result is the average execution velocity **over the entire RMF measurement interval**.

CPExpert compares the average execution velocity over the entire RMF measurement interval against the performance goal specified for the service class period. If the average execution velocity is less than the performance goal, CPExpert can conclude that the service class period did not achieve its performance goal for the RMF measurement interval. **This conclusion reveals a persistent problem**.

It is important to appreciate that the execution velocity goal may not be met during a number of Workload Manager policy adjustment intervals. This circumstance may not be detected when CPExpert analyzes RMF data as described above, since the average execution velocity is computed by CPExpert is based on an entire RMF measurement interval. CPExpert will detect a **persistent** problem, but cannot detect **periodic** problems with execution velocities being less than the performance goal.

CPExpert produces Rule WLM103 when CPExpert detects that a service class period did not meet its execution velocity goal for an entire RMF measurement interval. CPExpert reports the percent CPU Using samples, percent total waiting samples, the resulting execution velocity, and the primary and secondary causes of delay. Additionally, CPExpert computes the contribution that the primary and secondary causes of delay made to the address space delay.

CPExpert analyzes the following possible delays to service classes with an execution velocity goal[12]:

[12]Please see Section 4 (Chapter 3.3) for a description of these delays.

- **Denied CPU delay**
- **CPU Capping delay**
- **Swap-in delay**
- **MPL delay**
- **Page-in delay**
- **I/O delay**
- **Queue delay (Batch job initiator delay, TSO LOGON delay, or APPC request queue delay)**

The above causes of delay are analyzed by CPExpert in other rules.

For the purposes of identifying primary and secondary causes of response delay, CPExpert combines all auxiliary storage page-in delays into "page-in delay" to reflect the impact of auxiliary storage on response.

Notice that "CPU Using" is not included in the delays analyzed by CPExpert, as "CPU Using" is the **objective** of an execution velocity goal. Additionally, "Unknown" delay is not included in the delays analyzed by CPExpert, as "Unknown" delay is not included in the computation of execution velocity.

Each of the above causes of delay are analyzed by CPExpert in other rules.

The following example illustrates the output from Rule WLM103:

```
RULE WLM103:  SERVICE CLASS DID NOT ACHIEVE VELOCITY GOAL

   VEL40 (Period 1): Service class did not achieve its velocity goal
   during the measurement intervals shown below.  The velocity goal was
   40% execution velocity, with an importance level of 2.  The '% USING'
   and '%TOTAL DELAY' percentages are computed as a function of the average
   address space EXECUTING time (to exclude activity and delays not under
   WLM control). The 'PRIMARY,SECONDARY CAUSES OF DELAY' are computed as
   a function of the execution delay samples on the local system.

                         ------LOCAL SYSTEM--------
                           %    % TOTAL EXEC   PERF  PLEX PRIMARY,SECOND
   MEASUREMENT INTERVAL  USING   DELAY  VELOC  INDX   PI  CAUSES OF DELAY
   10:00-10:15,19AUG2003   9.1   16.9    35%   1.15  0.71 DENIED CPU(85%)
   10:15-10:30,19AUG2003   9.2   20.6    31%   1.30  0.72 DENIED CPU(69%)
   10:30-10:45,19AUG2003   8.1   17.4    32%   1.26  0.71 DENIED CPU(68%)
   10:45-11:00,19AUG2003   7.5   13.6    36%   1.13  0.68 DENIED CPU(64%)
```

Note that the % USING and %TOTAL DELAY percentages are computed as a function of the average address EXECUTING time. In the above example, the data shown for 10:00 indicates that the VEL40 service class was delayed for 16.9% of the time that it was executing on the local system. This view is of the time when the service class was under control of the WLM (that is, the percent *excludes* such things as IDLE samples and UNKNOWN samples, over which the WLM has no control).

While the service class was delayed (the 16.9% shown above), 85% of the 16.9% delay was due to being denied access to CPU. The 85% CPU delay was calculated as:

$$\textit{Percent CPU Delay} = \frac{\textit{R723CCDE}}{\textit{R723CTOT}}$$

Where

R723CCDE= CPU delay sample count

R723CTOT = Total general execution delay samples

These two views are important, because many analysts want to know how much WLM "manageable" delay[13] occurred to transactions in some online application (such as TSO) *while* transactions were being processed.

If IDLE and other delays not under WLM control were included in the "Total Delay", a very small number might be shown for the delay. This would be due to the fact that IDLE and other delays often account for a large percent of TSO time (for example). A small delay that included Idle time would be of little comfort to the user who might have experienced large delays waiting for transaction completion.

Notice that there is no "SECONDARY" cause of delay shown in the example output from Rule WLM103. CPExpert lists a SECONDARY cause of delay only if the delay is greater than the WLMSIG guidance variable.

Suggestion: There are no suggestions with this finding. CPExpert will continue analysis and other rules will be produced to provide more information.

[13]This specific example illustrates a more significant problem; namely, the Sysplex Performance Index is much less than 1 (indicating that, on a *sysplex* basis, the service class is exceeding its goal). As a consequence, the WLM might not take action to improve the performance of the service class period on the *local* system. This situation is discussed in Rule WLM140.