
Rule WLM221: Service Class was capped for discretionary goal management

Finding: CPExpert has determined that resource capping was a major cause of the service class not achieving its performance goal, but the service class had been capped for discretionary goal management.

Impact: The impact of this finding depends upon the amount of resource capping delay experienced by the service class.

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

- Rule WLM101: Service Class did not achieve average response goal
- Rule WLM102: Service Class did not achieve percentile response goal
- Rule WLM103: Service Class did not achieve execution velocity goal

Discussion: Resource capping is a way of controlling the distribution of CPU service to one or more service classes. Resource capping normally is implemented by defining "resource groups" to the Workload Manager. A resource group is simply a named set of two values: a minimum CPU service specification and a maximum CPU service specification. The specifications are in terms of **unweighted CPU service units** (that is, the CPU service coefficients are not applied to TCB nor SRB raw CPU service units).

The Workload Manager will attempt to provide the minimum CPU service to the resource group and will restrict the resource from using more than the maximum CPU service.

When the maximum CPU service specified in the resource group has been used, the Workload Manager marks "non-dispatchable" the TCBs and SRBs associated with the service classes assigned to the resource group.

This performance issue caused by normal resource capping is addressed by Rule WLM220. Rule WLM221 (this rule) addresses a slightly different issue.

A problem existed when using discretionary goals prior to OS/390 Version 2 Release 6: on systems in which 100% of the CPU was used by service class periods with performance goals, service class periods assigned a discretionary goal might never receive CPU service. This situation existed even though the service class periods with performance goals might be significantly **overachieving** their goals, since the Workload Manager would

never allow discretionary work to have a CPU dispatching priority equal to or higher than work with performance goals.

From one perspective, this algorithm is proper; discretionary work is defined as work that has no performance goal. However, most sites want the discretionary work eventually to be processed, even though it has no performance goal. Consequently, many sites removed the discretionary goal from work and assigned a performance goal to the work.

However, there are significant advantages to assigning a discretionary goal to work: work with a discretionary goal executes with the Mean-Time-To-Wait (MTTW) algorithm.

C Work assigned to a Mean-Time-To-Wait group competes within the Mean-Time-To-Wait group for access to the processor. Address spaces are assigned dispatching priority within the MTTW group, based upon their execution characteristics. Address spaces that execute a significant amount of CPU instructions between I/O operations are considered heavy CPU users. These heavy CPU users receive a lower dispatching priority within the MTTW group than do address spaces requiring less CPU processing between I/O operations.

C The philosophy behind assigning work to Mean-Time-To-Wait groups is to attempt to use as much of the overall computer system as possible. Dispatching relatively light CPU users ahead of relatively heavy CPU users ensures that the I/O complex will be used simultaneously with the CPU processor. Since both CPU and I/O are active simultaneously, more overall work will be accomplished by the computer system. This philosophy assumes, of course, that overall throughput is a major goal, rather than the turnaround of specific heavy CPU users. This philosophy is explicitly applicable to service class periods assigned a discretionary goal.

IBM addressed this problem in OS/390 Version 2 Release 6, by implementing the *discretionary goal management* algorithms .

With discretionary goal management, the Workload Manager identifies service class periods that have been assigned a performance goal and that are candidates for participation in discretionary goal management. Service class periods can participate in discretionary goal management if either of the following conditions apply:

C The service class period has a response goal greater than one minute. This condition does not apply to subsystem transaction service classes (e.g., CICS or IMS transaction service classes), since these service class periods do not include address spaces.

C The service class period has an execution velocity goal less than or equal to 30%.

The Workload Manager identifies candidate service class periods meeting either of the above conditions, that have **significantly** overachieved their performance goal. If discretionary work exists in the system, the Workload Manager may apply *internal resource capping* to the service class periods that are overachieving their performance goal. The internal resource capping operates similarly to the normal Resource Group capping described in Chapter 1.6 of this section, in that the Workload Manager will cap the address spaces for one or more cap slices. This capping restricts the amount of CPU service that can be used by address spaces in the capped service class period.

The Workload Manager may apply internal resource capping when the Performance Index is less than 0.7, and stops internal resource capping when the Performance Index is greater than or equal to 0.81. If a candidate service class period with a performance goal has multiple periods, later periods are selected for capping before earlier periods (that is, capping would potentially be applied to Period 2 before capping would be considered for Period 1).

The effect of the discretionary goal management algorithm is to allow discretionary work to receive CPU cycles when work with a performance goal would otherwise significantly overachieve its performance goal.

As the System Resources Manager takes its samples of the state of address spaces, it examines whether a dispatchable unit (TCB or SRB) is marked non-dispatchable because of a resource group maximum. Samples reflecting the resource group maximum are recorded by RMF in the SMF Type 72 delay samples, as CPU Capping Delay (R723CCCA).

CPExpert computes the percent of CPU Capping Delay for the service class, as a function of the overall execution of transactions executing in the service class.

C CPExpert produces Rule WLM220 if the percent of CPU Capping Delay for the service class is greater than the significance value specified in the **WLMSIG** guidance variable in USOURCE(WLMGUIDE) **and** the service class had been assigned to a Resource Group.

C CPExpert produces Rule WLM221 if the percent of CPU Capping Delay for the service class is greater than the significance value specified in the **WLMSIG** guidance variable in USOURCE(WLMGUIDE) and the service class had **NOT** been assigned to a Resource Group.

With Rule WLM221, CPExpert provides the total number of ending transactions in the RMF measurement interval, the total CPU service units consumed by the transactions, the average CPU service units per transaction, and the average percent resource capping delay to transactions active in the service class.

The following example illustrates the output from Rule WLM221:

RULE WLM221:SERVICE CLASS WAS CAPPED FOR DISCRETIONARY GOAL MANAGEMENT				
Service Class BATCHLO (Period 2) was delayed waiting for CPU resource capping. This means that a TCB or SRB in the Service Class was marked non-dispatchable because the Resource Group maximum was being enforced. The service class was not assigned to a Resource Group, but the Workload Manager implemented internal resource capping as a part of discretionary goal management. Normally, this will not be a concern (as the WLM will not implement internal resource capping unless the service class period is over-achieving its goal).				
	TOTAL	TOTAL CPU	AVERAGE CPU	AVG %
MEASUREMENT INTERVAL	TRANS	SERVICE UNITS	SERVICE UNITS	DELAY
15:00-15:16,01MAR1994	8	36,892	4611	14.4

Suggestion: This finding normally should not be produced, as explained in the above discussion; the Workload Manager will not select a service class period with a performance goal for internal resource group capping unless the service class period is significantly overachieving its performance goal.

Reference: MVS Planning: Workload Management

- MVS/ESA(SP 5): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V1R1): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V1R2): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V1R3): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R4): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R5): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R6): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R7): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R8): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R9): Chapter 8: Defining Service Classes and Performance Goals
- OS/390 (V2R10): Chapter 8: Defining Service Classes and Performance Goals
- z/OS (V1R1): Chapter 8: Defining Service Classes and Performance Goals
- z/OS (V1R2): Chapter 8: Defining Service Classes and Performance Goals
- z/OS (V1R3): Chapter 8: Defining Service Classes and Performance Goals
- z/OS (V1R4): Chapter 8: Defining Service Classes and Performance Goals

"Pop the Hood on Workload Manager", Steve Grabarits and Gail Whistance
(IBM Corporation Workload Manager developers), Session 2513, SHARE
Technical Conference, August 1998.