Rule WLM210: Average Server CPU use per transaction is higher than goal

Finding: CPExpert has determined that the average server CPU time per transaction was higher than the response goal for the service class.

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

Logic flow: The following rule causes this rule to be invoked:
Rule WLM120: Significant transaction time was in Active state

Discussion: When CPExpert produces Rule WLM104 or Rule WLM105 to indicate that a subsystem service class did not achieve its performance goal, the logic of these rules tries to identify the cause of the delay. The cause of the delay initially is analyzed from the "served" service class view. Rule WLM120(series) and Rule WLM130(series) describe the results from this analysis.

After analyzing the subsystem transaction delays, CPExpert identifies the service classes which serve the transactions. The subsystem transactions typically are CICS transactions, and the servers are the CICS regions. Alternatively, the transactions could be IMS transactions and the servers could be the IMS control regions or transaction processing regions.

Address spaces executing in the system can be in a variety of states from the perspective of the Workload Manager: using the CPU, delayed for an identifiable reason, or delayed for some unknown reason.

The System Resources Manager (SRM) periodically samples the state of each address space in each service class. These samples are accumulated into variables which are recorded by RMF in the "Service Class Period Data Section" of SMF Type 72 (Subtype 3) records. Please see Section 4 for a discussion of these states and the sampling process.

CPExpert produces Rule WLM120 when a significant cause of delay to a subsystem transaction was that the transaction was in Active state. The Active state indicates that a task was executing on behalf of the transaction, from the perspective of CICS or IMS. CPExpert analyzes the CPU requirements of the server service class to determine whether the transaction required a significant amount of CPU time.

CPU usage and other resource requirements are not contained in the SMF Type 72 records which describe the subsystem transaction service class.

These subsystem transaction service classes are not address spaces, but are logical groupings of transactions. Resource information is **not** recorded by SMF for the transactions, but the resource information **is** recorded for the address spaces (the servers) providing service to the service classes. Consequently, CPExpert analyzes the resource requirements of the server service classes.

CPExpert analyzes the amount of CPU time used by transactions by the following process:

- CPExpert first computes the number of samples which found an address space executing in the service class. This is done by summing CPU Using samples (R723CCUS), Total Wait samples (R723CTOT), and Unknown Delay samples (R723CUNK). The result is titled "EXSAMP" in the code.

- CPExpert divides the number of CPU Using samples (R723CCUS) by the EXSAMP value, to yield the percent of execution samples in which the SRM found an address space was using the CPU. The average transaction response time is multiplied by the resulting percentage to yield the amount of time when the average transaction was using the CPU.

- Server service classes might serve multiple subsystem service classes. For example, a CICS region (the server) might provide service to a number of service classes which describe different CICS transactions. If the server provides service to multiple service classes, the above technique automatically "pro-rates" the CPU requirements of the server. This automatic "pro-rating" is possible because the sampling process of the SRM is independent of the transactions executing.

CPExpert produces Rule WLM210 if the average CPU time per transaction is higher than the performance goal for the transaction service class. With Rule WLM210, CPExpert shows the percent of total service provided by the server to the transaction service class missing its performance goal. This value is computed by dividing the number of times an address space in the server provided service to the transaction service class, by the total number of times the server provided service to all transaction service classes.

If CPExpert is analyzing served transactions from the Execution Phase view, and if no transactions ended Execution Phase, CPExpert will produce “???” in the TRANS column. Otherwise, CPExpert will print the number of transactions that completed Execution Phase.

If CPExpert is analyzing served transactions from the Begin_to_end (BTE) Phase view, and if no transactions ended in the BTE Phase, CPExpert will produce "???" in the TRANS column. Otherwise, CPExpert will print the number of transactions that completed the BTE Phase.

The following example illustrates the output from Rule WLM210:

```
RULE WLM210: AVERAGE SERVER CPU USE PER TRANSACTION IS HIGHER THAN GOAL

  The average CPU time per transaction by the server (CICSRGN) was higher
  than the response goal for Service Class CICUSRTX.  If CICSRGN provided
  service to more than one service class, CPExpert prorated the CPU time
  based on the number of times that CICSRGN provided service to CICUSRTX.
  Using these calculations, the average transaction used more CPU time
  than the response goal of CICUSRTX.  This situation applies to the
  following RMF measurement intervals:

                                TOTAL      AVG SERVER CPU         PCT
  MEASUREMENT INTERVAL          TRANS      TIME PER TRANS       SERVICE
  13:07-13:12,21JUN1994        14,307       0:00:00.836          99.5
  13:17-13:22,21JUN1994        14,314       0:00:00.834          99.7
```

Suggestion; MVS cannot achieve the specified response goal for the service class unless the CPU requirements of the average transaction can be reduced.

CPExpert suggests that you consider the following actions:

- Perform a "reality" check on the finding from CPExpert by examining the "Response Time Distribution" produced by Rule WLM108 or Rule WLM109 (one of these rules will be produced depending upon the nature of the service class and performance goal). Determine whether most transactions missed the response objective or whether a few transactions **significantly** missed the response objective. If only a few transactions **significantly** missed the response objective, it is likely that these transactions skewed the findings.

  If you find that some transactions skewed the findings, you may wish to consider other alternatives:

  - If you can identify the transactions, perhaps you can use Workload Categorization to place the transactions into a different service class. You may wish to specify a different importance and different performance goal for this new service class.

  - If you have specified an **average response goal** for the service class, perhaps you can change the goal to a **percentile response goal**. With a percentile goal, the Workload Manager would not be as concerned about the few transactions which used significantly more resources and consequently skewed the average response. Rather,

the Workload Manager would base its workload management decisions on the percent of transactions which met the response goal.

- Review your performance goal for the transactions served by the service class, to determine whether the response goal is correct.

- Review the application processing the transactions, to determine whether the application code can more efficiently use the CPU. If the application code can be made more efficient, less CPU time will be required to process the transactions.

- Review the CPU requirements of the server (either the CICS region or the IMS region).

  If the server is CICS, you should execute the CPExpert CICS Component against the CICS region to identify performance improvement opportunities. If you have not licensed the CPExpert CICS Component, you should follow the "Processor Cycles Checklist" in IBM's *CICS Performance Guide*.

  If the server is an IMS region, IBM suggests the following actions to reduce CPU time used by the IMS region[1]:

  > "The total number of machine instructions that are executed to process a transaction, including system services, IMS services, and the application program itself, has a direct bearing on throughput. The accumulation of executed instructions is termed the path length. The actions suggested in the previous IMS Options section all contribute to the minimization of path length.
  > "Avoid the regular use of traces such as the DL/I Call Image Capture and other traces invoked by the /TRACE command. These are specified as parameters on the OPTIONS statement in the DFSVSMxx member of IMS.PROCLIB.
  >
  > "Do not run the IMS Monitor (DFSMNTR0), except for 10- to 20-minute preplanned intervals.
  >
  > "In a real-storage constrained system, the most effective way to reduce path length is to minimize paging. Minimal pools contribute to minimize paging by eliminating costly scanning of directories or buffers which might have to be paged in before they can be read. If virtual storage requirements are reduced:
  >
  > - A minimal PSB pool minimizes buffer searching.

[1]Source: *IMS/ESA Version 4: System Administration Guide*, Section 7.2.6 Minimizing Path Length (BookManager document).

- A tuned database pool minimizes buffer searching; a larger database pool costs more in path length and might not reduce I/O.

- A tuned message queue pool minimizes buffer searching; a larger pool reduces IMS message queue I/O but at the expense of a higher processor cycles per queue pool operation.

- The same applies to the message format pool as to the message queue pool."

- If none of the above options are applicable, and if this service class is very important, you may wish to consider running the application on a more powerful processor.

  Note that simply increasing the Importance specified to the Workload Manager, or adding more logical processors (in an LPAR environment) will not resolve the problem with the service class not achieving its response goal. Transactions are delayed because the server service class is using the CPU, not because the server is denied access to the CPU[2].

[2]Although other rules may show that transactions also are denied access to the CPU, Rule WLM210 reports that subsystem transactions are delayed because of the amount of CPU use by the server service class.

