I am not an SQL expert, which everyone should know since I only recently discovered how to use VIEWs to make my tasks a bit easier. Any SQL I write and share should be reviewed with that in mind. These queries are also written based on the column names assigned by MQSMFCSV, which will be different is using another SMF interpreter. This is also considered open source and should be treated as such. It is presented as-is and there is no implied or stated support.

Years ago, I began using the WQ records to provide summaries of queue level activity I briefly looked into the breakdown of Coupling facility calls that were being made. I was told by people with the correct accent that I really did not need to know about those fields, and for years I did not. However, as those people who assured me that I didn't need to know are gone, I ended up having to look into these fields and found that yes we do need examine those records in that level of detail because it led to an explanation for some behavior that has been observed several times now by customers that are heavy users of shared queues in various patterns. We also uncovered some gaps in the data collection that are being addressed by the development lab. This is complex performance problem determination, looking into things that are not observed everything is running well.

If you have not read the articles on Task Records and how they are associated, Using Views, and symptoms that led to finding this issue (which has already been applied to some other customer data), I'd recommend a quick overview of those first. The SQL development took several iterations to get meaningful information. We were fortunate to have customer data from days where the processing flowed normally and where they had problems. How often have I said, "It is hard to spot abnormal when you don't know what normal looks like"? More than I can count certainly. In this case it was vital to finding the issues.

First it was troubling that the overall volume of requests was not noticeably higher on some of the days where there were issues reported. That's unusual, as many of the problems we see are related to volume changes. When we started analyzing the traffic in the queues themselves, from the Task associated WQ records, we did find changes in volumes on individual queues. We also found that the overall mix of messages was skewed towards one queue on one structure during the problem periods. And we could see after the individual queue analysis that one queue had an extremely low valid get percentage when those periods began. What was also interesting, was that we could see the percentage of valid gets fluctuate on this queue even in 'good' periods at times almost dropping to the same level as when there were noted problems.

We then started looking at the types of MQGETs being done against the shared queues in this QSG. That led to looking at the actual CF calls being made, and that made all the difference. The query used was just for the GET processing being done. When we tried to combine the PUT processing into the same query we had far too many columns to be contained in a normal spreadsheet. As it turned out, we did not need all of the calls to the CF that the MQGETs resolved into, so we did simplify the query to look for the 'READLIST' and 'GETMOVE' CF requests in our comparisons.

This shorter version of the 'What calls are being made' query looks like what is shown below, and is included in this git repository as CFGETsBreakDown.txt.

What we found was simple, the volume of requests skewed toward a particular type of processing when the issue began – that was to do gets by a message selector. This is more work for the queue manager as it has to read thru all the messages on the queue looking for the right match. Unlike private queues where there is a 'messages skipped' count there is no equivalent for shared queues. Also the use of message selectors only shows up in the WQ records when the queue is opened, and as these were very long running tasks at first glance it looked like request to the problem structure and queue were not using selectors. It was only when we looked at the actual CF calls that we could see that the processing for a particular queue on a specific structure that was different. Each MQGET was resolving into anywhere from a few (during non-problem periods) to several thousand 'READLIST' requests (during problem periods) to the CF structure. This was because as the queue got deeper more READLISTs were required to scroll thru all the messages. To make matters worse the applications issuing the MQGETs had a very short timeout (in seconds), but the messages themselves were not expiring for minutes. That meant that messages were building up on the queue that would never be retrieved while at the same time new messages were being put. The solution was to alter the processing to use an indexable field rather than a message selector, to set a message expiration closer to the value used for the MQGET, and to set the queue manager expiration interval to run more frequently on all queue managers in the QSG. The application changes took more time to implement, so to help prevent the issue from impacting other application and other processes that were part of this application, the queue was moved to its own CF structure so there would be less competition for resources and the queue manager expiry interval was set on all queue managers to help clean up the queue more frequently.

It was also interesting to discover the issue was also present during period where there had not been a detected problem. There was a specific volume of failing MQGETs before the impact was detected by the application and users – when we reviewed some of the 'good' processing periods we also found failing MQGETs, and at times it seemed to border on the problem threshold. This was a classic problem waiting to happen.

```
EXPORT TO "E:\customer\Query_Results\GETTasksByQ.csv" OF DEL MODIFIED BY COLDEL, DECPT.

SELECT

CHAR(WTID.DATE) AS DATE1,

WTID.TIME AS TIME1,

WTID.LPAR AS LPAR,

WTID.QMGR AS QMGR,

WTID.WTAS_CORRELATOR AS CORRELID,

WTID.CHANNEL_NAME AS CHL_NAME,

WTID.Channel_Connection_Name AS CONNECTION_NAME,

CHAR(WTAS.Commit_Count) AS COMMIT_COUNT,
```

```
CHAR(WTAS.Commit_ET_us) AS COMMIT_ET,
CHAR(WTAS.Commit_CT_us) AS COMMIT_CPU,
CHAR(WTAS.Backout_Count) AS BACKOUT_COUNT,
CHAR(WTAS.Backout_ET_us) AS BACKOUT_ET,
CHAR(WTAS.Backout_CT_us) AS BACKOUT_CPU,
CHAR(WTAS.START_TIME_DATE) AS TASK_START_DATE,
CHAR(WTAS.START_TIME_TIME) AS TASK_START_TIME,
CHAR(CF_STE_CALL_COUNT) AS SINGLE_ENTRY_CALL_COUNT,
CHAR(CF_STM_CALL_COUNT) AS MULTIPLE_ENTRY_CALL_COUNT,
CHAR(CF_STE_REDRIVE_COUNT) AS SINGLE_ENTRY_REDRIVES,
CHAR(CF_STM_REDRIVE_COUNT) AS MULTIPLE_ENTRY_REDRIVES,
CHAR(CF_STE_ELAPSED_US) AS SINGLE_ENTRY_ET,
CHAR(CF_STM_ELAPSED_US) AS MULTIPLE_ENTRY_ET,
WQ.Open_Name AS OPEN_NAME,
WQ.Base_Name AS BASE_NAME,
CHAR(WQ.GET_COUNT) AS GET_COUNT,
CHAR(WQ.TOTAL_VALID_GETS) AS VALID_GET_COUNT,
CHAR(WQ.GET_CT_US)   AS GET_CPU,
CHAR(WQ.GET_ET_US) AS GET_ET,
CHAR(GET_DEST_ANY_COUNT) AS MQGET_ANY_COUNT,
CHAR(GET_DEST_SPECIFIC_COUNT) AS MQGET_SPECIFIC_COUNT,
CHAR(MAX_DEPTH) AS MAX_QUEUE_DEPTH,
CHAR(SELECT_COUNT) AS SELECTOR_COUNT,
CHAR(SELECT_MAX_LENGTH) AS SELECTOR_MAX_LEN,
CHAR(CFCOUNT_GET_READLIST) AS CFREADLIST_COUNT,
CHAR(CFSYNC_GET_READLIST) AS CFSYNC_READLIST_COUNT,
CHAR(CFSYNC_GET_READLIST_ET) AS CFSYNC_READLIST_ET,
CHAR(CFASYNC_GET_READLIST) AS CFASYNC_READLIST_COUNT,
CHAR(CFASYNC_GET_READLIST_ET) AS CFASYNC_READLIST_ET,
CHAR(CFCOUNT_GET_MOVE) AS CFGETMOVE_COUNT,
CHAR(CFSYNC_GET_MOVE) AS CFSYNC_GETMOVE_COUNT,
CHAR(CFSYNC_GET_MOVE) AS CFSYNC_GETMOVE_ET,
```

```
CHAR(CFASYNC_GET_MOVE) AS CFASYNC_GETMOVE_COUNT,
CHAR(CFASYNC_GET_MOVE) AS CFASYNC_GETMOVE_ET,
      '2' AS Row_ID
FROM QSGQSGA_WTID AS WTID, QSGQSGA_WTAS AS WTAS, QSGQSGA_WQ  WQ
WHERE ( WTID.WTAS_Correlator =  WTAS.Correl
    AND WQ.CORRELATION = WTAS_Correlator
    AND WTID.DATE = WTAS.DATE
    AND WQ.Date = WTAS.Date
     ND WQ.TIME = WTAS.TIME
    AND WTAS.TIME = WTID.TIME
    AND WTAS.QMgr = WTID.Qmgr
    AND WQ.QMGR  = WTAS.QMgr
    AND WQ.CF_STRUCTURE <> '       '
    AND WQ.GET_COUNT > 0     )
UNION
SELECT
' Date ',
' Time ',
' LPAR ',
' QMGR ',
' Task Correlation ID  ',
' Channel Name ',
' Channel Conection Name ',
' Commit Count ',
' Commit Elapsed Time ',
' Commit CPU Time ',
' Backout Count ',
' Backout Elapsed Time ',
' Backout t CPU Time ',
' WTAS Start Date ',
' WTAS Start Time ',
' WTAS Single Entry CF Call Count ',
```

```
' WTAS Multiple Entry CF Call Count ',
' WTAS Single Entry Redrive Count ',
' WTAS Multiplee Entry Redrive Count',
' WTAS Single Entry Elapsed Time ',
' WTAS Multiple Entry Elapsed Time ',
' Queue Open Name ',
' Queue Base Name ',
' Get Count ',
' Valid Get Count ',
' Get CPU ',
' Get Elapsed Time ',
' Get Destructive Any ',
' Get Destrcutive Specific ',
' Maximum Queue Depth ',
' Get Selector Count ',
' Get Selector Max Length ',
' Get Readlist Count ',
' Get Sync Readlist Count ',
' Get Sync Readlist Elapsed Time ',
' Get ASync Readlist Count ',
' Get ASync Readlist Elapsed Time ',
' Get Getmove Count ',
' Get Sync Getmove Count ',
' Get Sync Getmove  Elapsed Time ',
' Get ASync Getmove Count ',
' Get ASync Getmove Elapsed Time ',
     '1' AS Row_ID
FROM SYSIBM.SYSDUMMY1 ORDER BY Row_ID
;
```